# Graph Neural Networks

Natalia Dubljevic
Biomedical Engineering
Schulich School of Engineering

W2024

UNIVERSITY OF CALGARY

# Outline

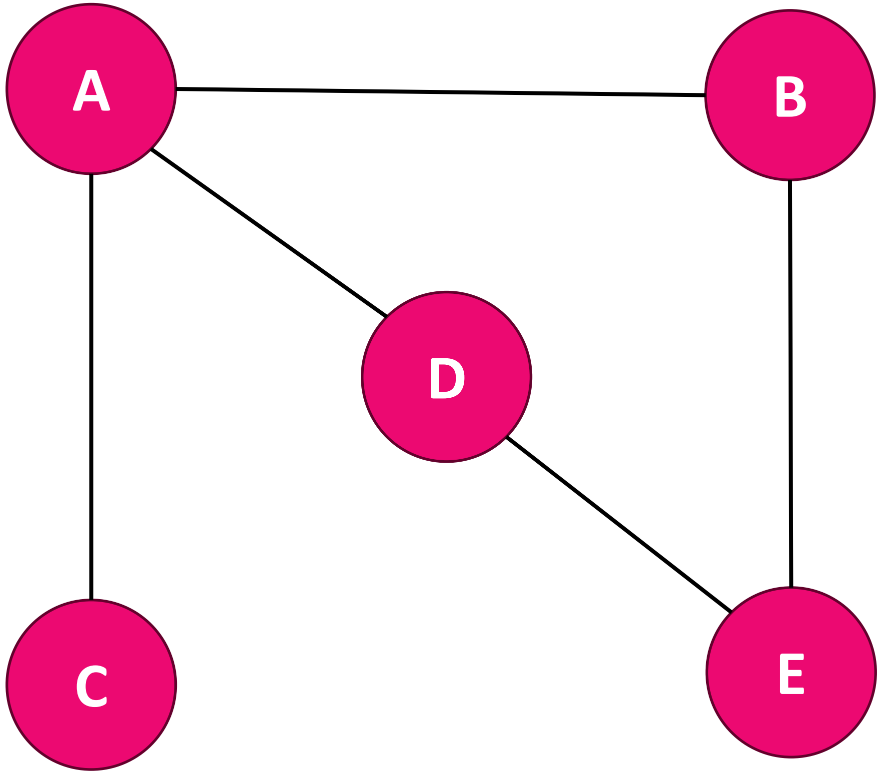
- Learning Goals
- Graph neural networks
  - What is graph-structured data
  - What sorts of problems can we solve using graph neural networks
  - How are graph neural networks designed
- Summary
- Tutorial

# Learning Goals

- Understand graph structured data and its differences from data we've seen so far
- What are some applications of GNNs
- Some of the core concepts of GNN implementation such as
  - Message passing
  - K-hop neighbourhood

# What is a graph anyways?

- Graphs have **nodes** and **edges**
- Edges represent the relationship between any two nodes






UNIVERSITY OF CALGARY

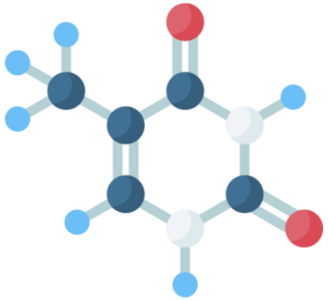
# Graph structured data

Graphs can be used to represent data such as:

## Social networks

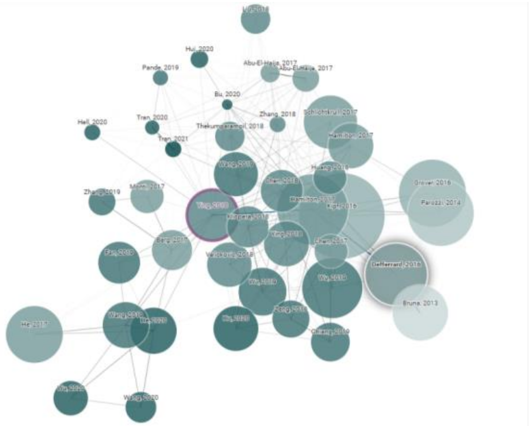
[University of Kentucky]

## Molecules



[flaticon.com]

## Citation Maps

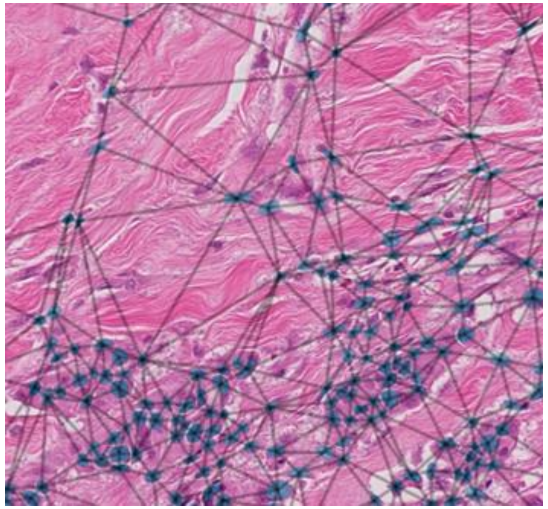


[connectedpapers.com]

## Transit


[translink.ca]

## Tissues



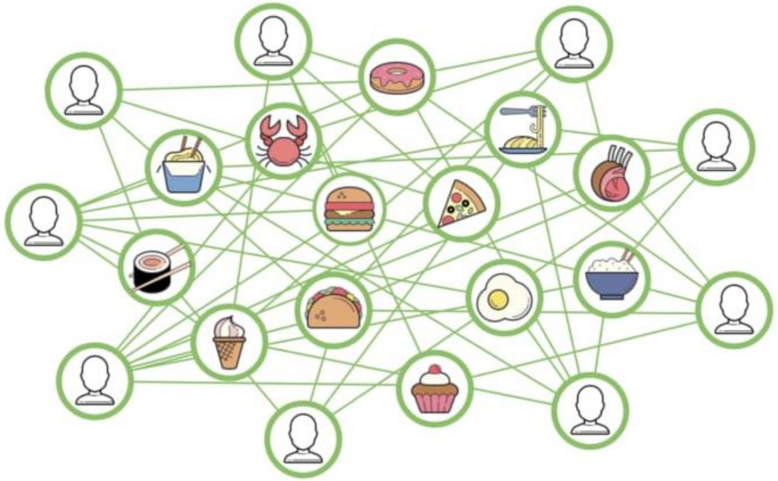


UNIVERSITY OF CALGARY

# Graph structured data

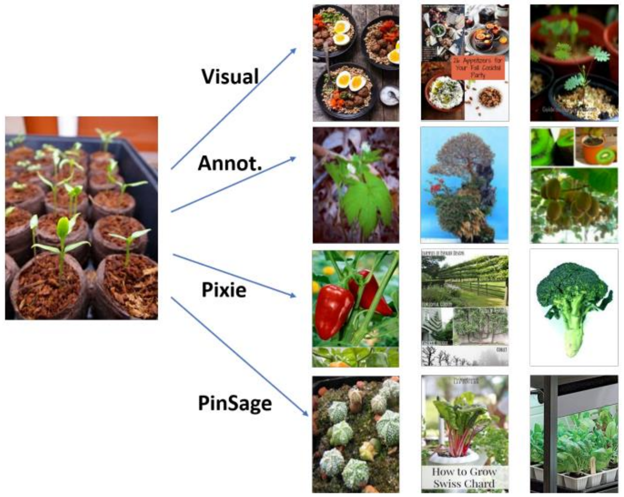
GNNs aren't just for niche applications-- they're trendy in industry too!

**Uber Eats**



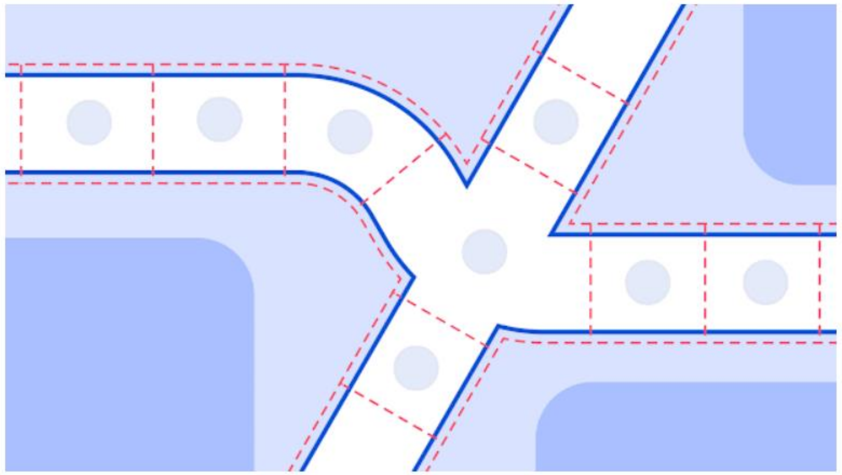
[uber.com]

**Pinterest**




[Ying et al. 2018]

**Google maps**

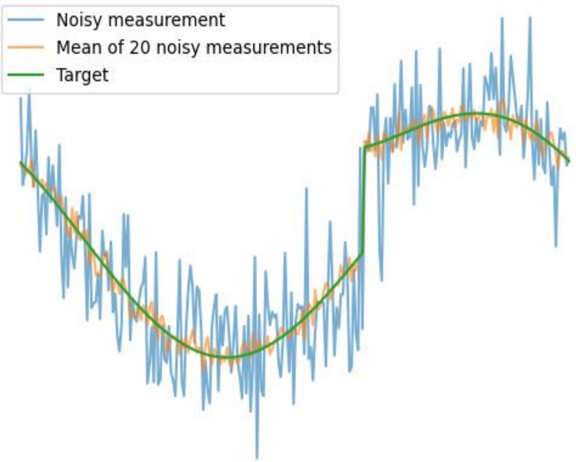


[deepmind.google.com]

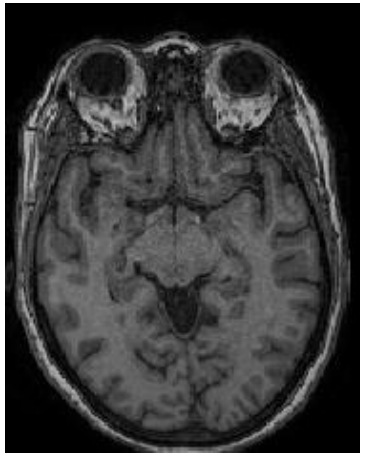
UNIVERSITY OF CALGARY

# What we've seen so far

## Signals




## Images



## Text

Love looks not with the eyes, but with the mind; And therefore is wing'd Cupid painted blind. Nor hath love's mind of any judgment taste; Wings and no eyes figure unheedy haste: And therefore is love said to be a child, Because in choice he is so oft beguil'd.

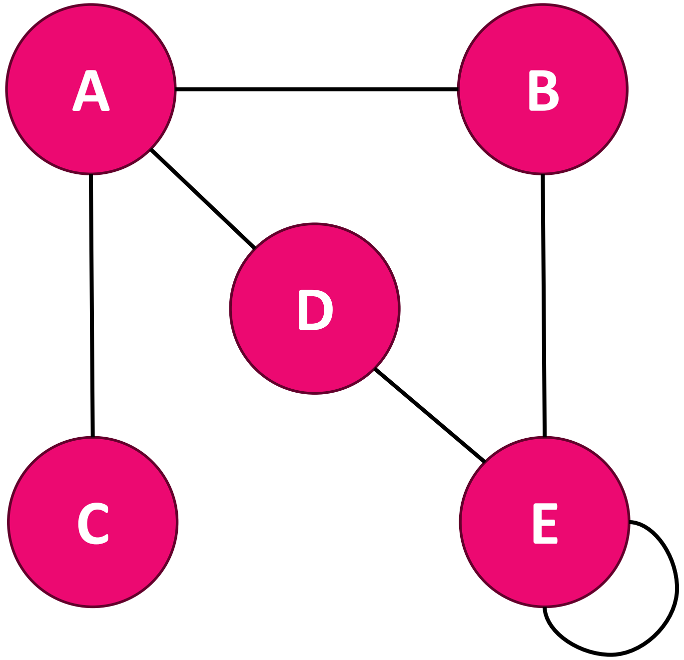
## Tabular

| | age | workclass | fnlwgt | education | education-num | marital-status | occupation | relationship |
|---|---|---|---|---|---|---|---|---|
| 0 | 39 | State-gov | 77516 | Bachelors | 13 | Never-married | Adm-clerical | Not-in-family |
| 1 | 50 | Self-emp-not-inc | 83311 | Bachelors | 13 | Married-civ-spouse | Exec-managerial | Husband |
| 2 | 38 | Private | 215646 | HS-grad | 9 | Divorced | Handlers-cleaners | Not-in-family |
| 3 | 53 | Private | 234721 | 11th | 7 | Married-civ-spouse | Handlers-cleaners | Husband |
| 4 | 28 | Private | 338409 | Bachelors | 13 | Married-civ-spouse | Prof-specialty | Wife |

- All of these can be represented as graphs!
- Graphs are one of the most general data structures

# A bit more on graphs

- We can represent graphs as an adjacency matrix

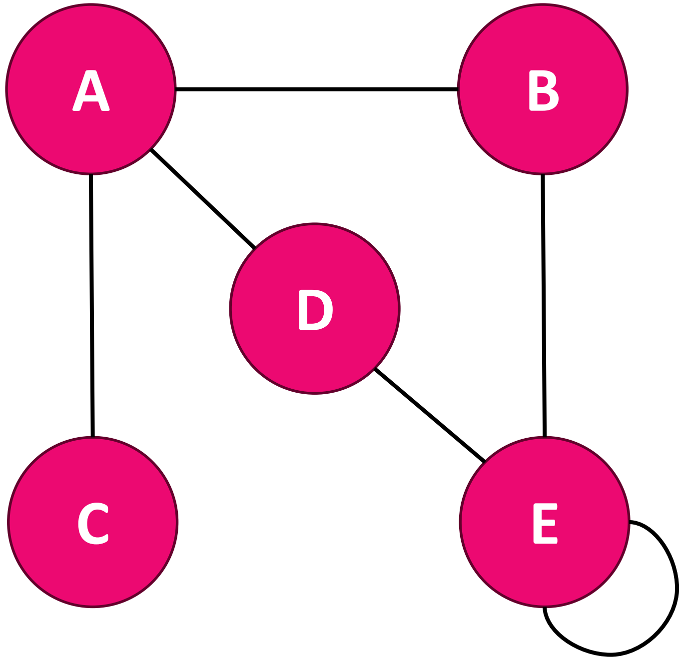



| | A | B | C | D | E |
|---|---|---|---|---|---|
| A | 0 | 1 | 1 | 1 | 0 |
| B | 1 | 0 | 0 | 0 | 1 |
| C | 1 | 0 | 0 | 0 | 0 |
| D | 1 | 0 | 0 | 0 | 1 |
| E | 0 | 1 | 0 | 1 | 1 |

# A bit more on graphs

- We can represent graphs as an adjacency matrix
- Edges can be weighted and/or directional

**Weighted edges**

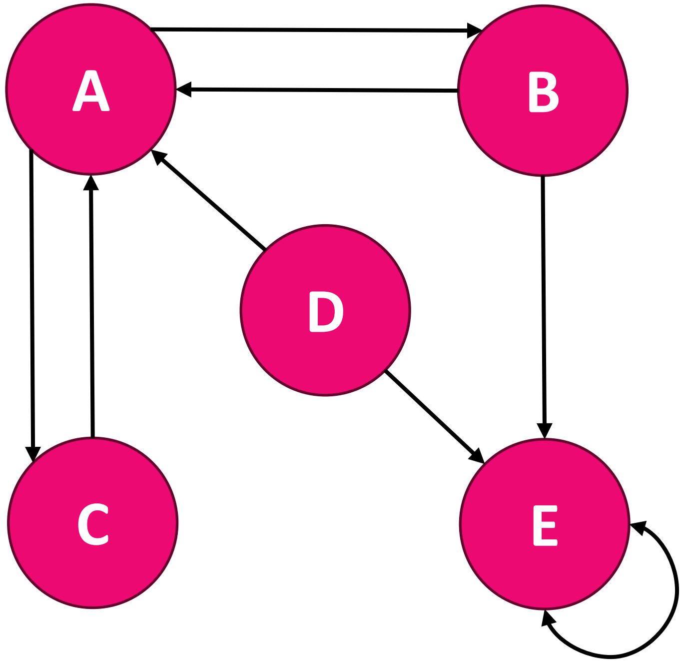



| | A | B | C | D | E |
|---|---|---|---|---|---|
| A | 0 | 0.4 | 1 | 0.2 | 0 |
| B | 0.4 | 0 | 0 | 0 | 1 |
| C | 1 | 0 | 0 | 0 | 0 |
| D | 0.2 | 0 | 0 | 0 | 1 |
| E | 0 | 1 | 0 | 1 | 1 |

# A bit more on graphs

- We can represent graphs as an adjacency matrix
- Edges can be weighted and/or directional

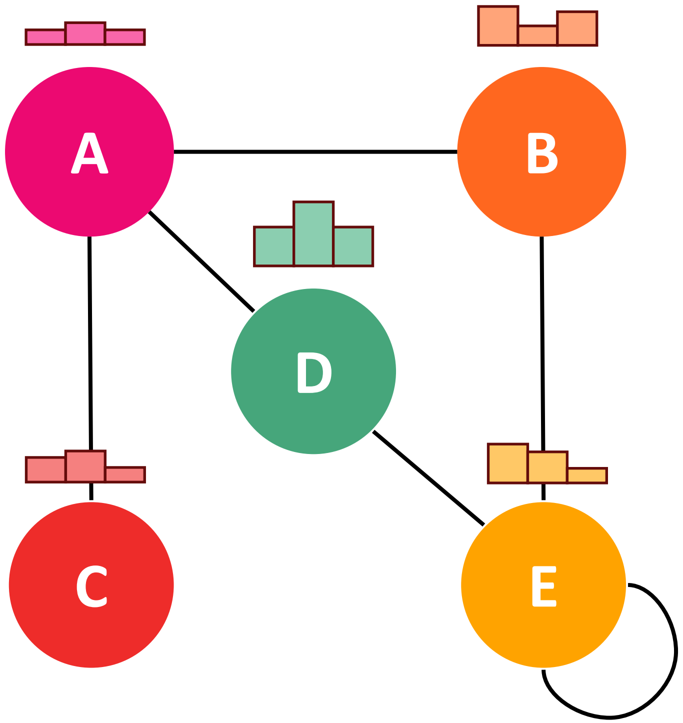
**Directional edges**




| | A | B | C | D | E |
|---|---|---|---|---|---|
| A | 0 | 1 | 1 | 0 | 0 |
| B | 1 | 0 | 0 | 0 | 1 |
| C | 1 | 0 | 0 | 0 | 0 |
| D | 1 | 0 | 0 | 0 | 1 |
| E | 0 | 1 | 0 | 1 | 1 |

# A bit more on graphs

- We can represent graphs as an adjacency matrix
- Edges can be weighted and/or directional
- Can have features related to nodes, edges, or graph as a whole
  - Node features most common




| | A | B | C | D | E |
|---|---|---|---|---|---|
| A | 0 | 1 | 1 | 1 | 0 |
| B | 1 | 0 | 0 | 0 | 1 |
| C | 1 | 0 | 0 | 0 | 0 |
| D | 1 | 0 | 0 | 0 | 1 |
| E | 0 | 1 | 0 | 1 | 1 |

# Working with sparse tensors

- Since they are often sparse, adjacency matrices are commonly represented as a list of edges

| | A | B | C | D | E |
|---|---|---|---|---|---|
| A | 0 | 1 | 1 | 1 | 0 |
| B | 1 | 0 | 0 | 0 | 1 |
| C | 1 | 0 | 0 | 0 | 0 |
| D | 1 | 0 | 0 | 0 | 1 |
| E | 0 | 1 | 0 | 1 | 1 |

# Working with sparse tensors

- Since they are often sparse, adjacency matrices are commonly represented as a list of edges

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 2 & 1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 & 1 \\ 4 & 0 & 1 & 0 & 1 & 1 \end{array}$$

# Working with sparse tensors

- Since they are often sparse, adjacency matrices are commonly represented as a list of edges

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 2 & 1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 & 1 \\ 4 & 0 & 1 & 0 & 1 & 1 \end{array}$$

$$\begin{pmatrix} (0, 1) \\ \\ \\ \\ \\ \end{pmatrix}$$



UNIVERSITY OF CALGARY

# Working with sparse tensors

- Since they are often sparse, adjacency matrices are commonly represented as a list of edges

$$
\begin{array}{c|ccccc}
 & 0 & 1 & 2 & 3 & 4 \\
\hline
0 & 0 & 1 & 1 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 \\
2 & 1 & 0 & 0 & 0 & 0 \\
3 & 1 & 0 & 0 & 0 & 1 \\
4 & 0 & 1 & 0 & 1 & 1
\end{array}
\qquad
\begin{pmatrix}
(0, 1) \\
(0, 2) \\
\\
\\
\\
\end{pmatrix}
$$

# Working with sparse tensors

- Since they are often sparse, adjacency matrices are commonly represented as a list of edges

$$
\begin{array}{c|ccccc}
 & 0 & 1 & 2 & 3 & 4 \\
\hline
0 & 0 & 1 & 1 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 \\
2 & 1 & 0 & 0 & 0 & 0 \\
3 & 1 & 0 & 0 & 0 & 1 \\
4 & 0 & 1 & 0 & 1 & 1
\end{array}
\qquad
\begin{pmatrix}
(0, 1) \\
(0, 2) \\
(0, 3) \\
\\
\\
\\
\end{pmatrix}
$$

# Working with sparse tensors

- Since they are often sparse, adjacency matrices are commonly represented as a list of edges

$$
\begin{array}{c|ccccc}
 & 0 & 1 & 2 & 3 & 4 \\
\hline
0 & 0 & 1 & 1 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 \\
2 & 1 & 0 & 0 & 0 & 0 \\
3 & 1 & 0 & 0 & 0 & 1 \\
4 & 0 & 1 & 0 & 1 & 1
\end{array}
\qquad
\begin{pmatrix}
(0, 1) \\
(0, 2) \\
(0, 3) \\
(1, 0) \\
\\
\\
\end{pmatrix}
$$

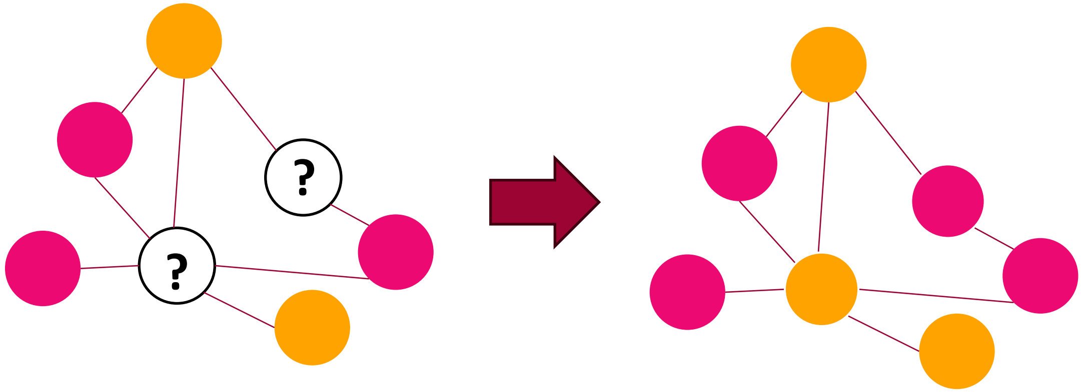
# Working with sparse tensors

- Since they are often sparse, adjacency matrices are commonly represented as a list of edges

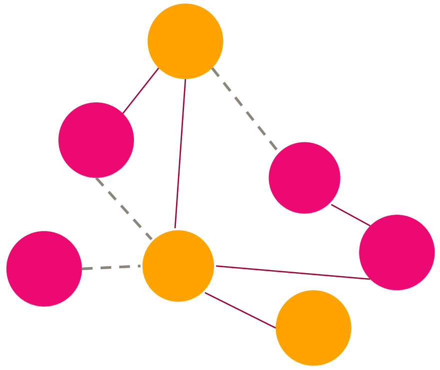
$$
\begin{array}{c|ccccc}
 & 0 & 1 & 2 & 3 & 4 \\
\hline
0 & 0 & 1 & 1 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 \\
2 & 1 & 0 & 0 & 0 & 0 \\
3 & 1 & 0 & 0 & 0 & 1 \\
4 & 0 & 1 & 0 & 1 & 1
\end{array}
\qquad
\begin{pmatrix}
(0,1) \\
(0,2) \\
(0,3) \\
(1,0) \\
(1,4) \\
\vdots
\end{pmatrix}
$$

# Tasks

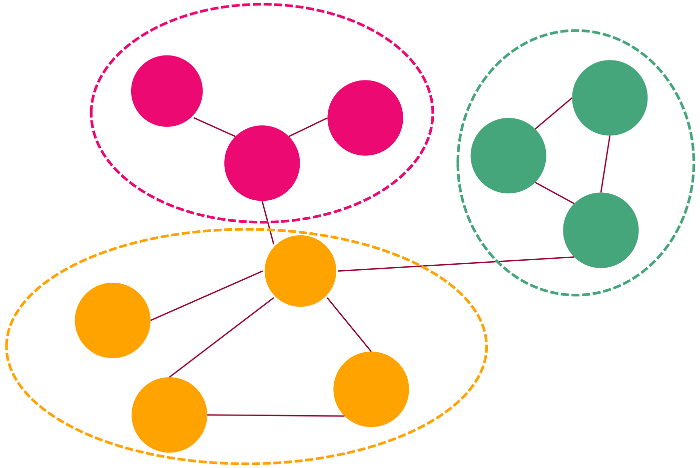
## Node Classification



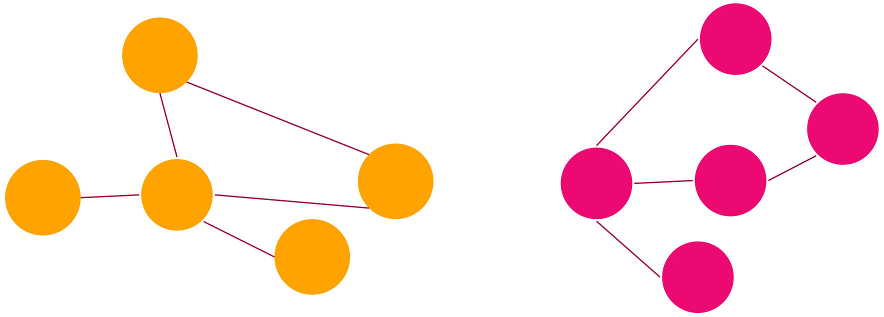

## Edge prediction



## Clustering



## Graph Classification

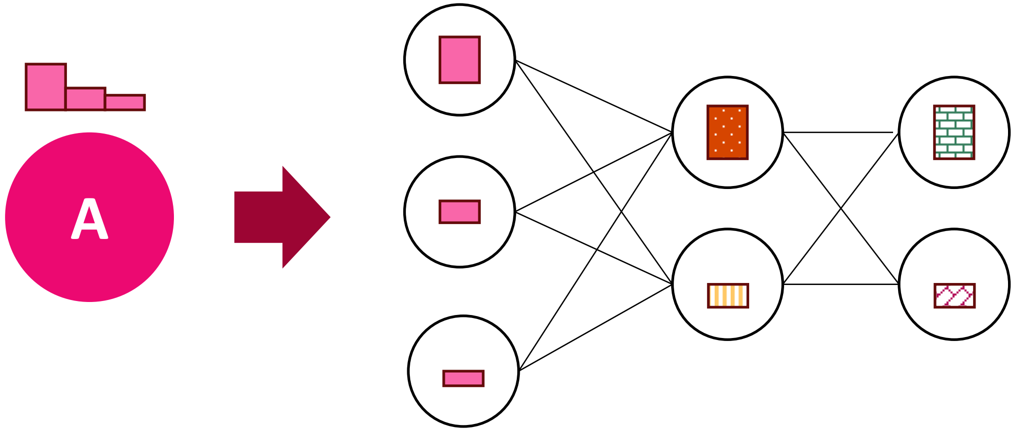




UNIVERSITY OF CALGARY

# Prelude to message passing

- How do we create models that can accommodate graph-structured data?
- Naïve approach: Feed a node and its features into an FCNN



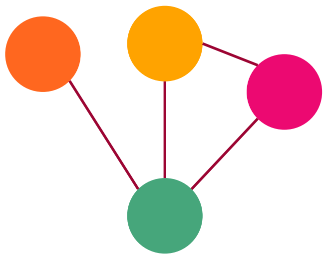

- **Question:** Are there any issues with this approach?



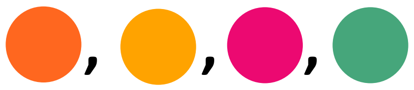
UNIVERSITY OF CALGARY

# Prelude to message passing

- Unlike standard supervised learning, we do not have independent points!
  - Our nodes are connected to each other in unique ways

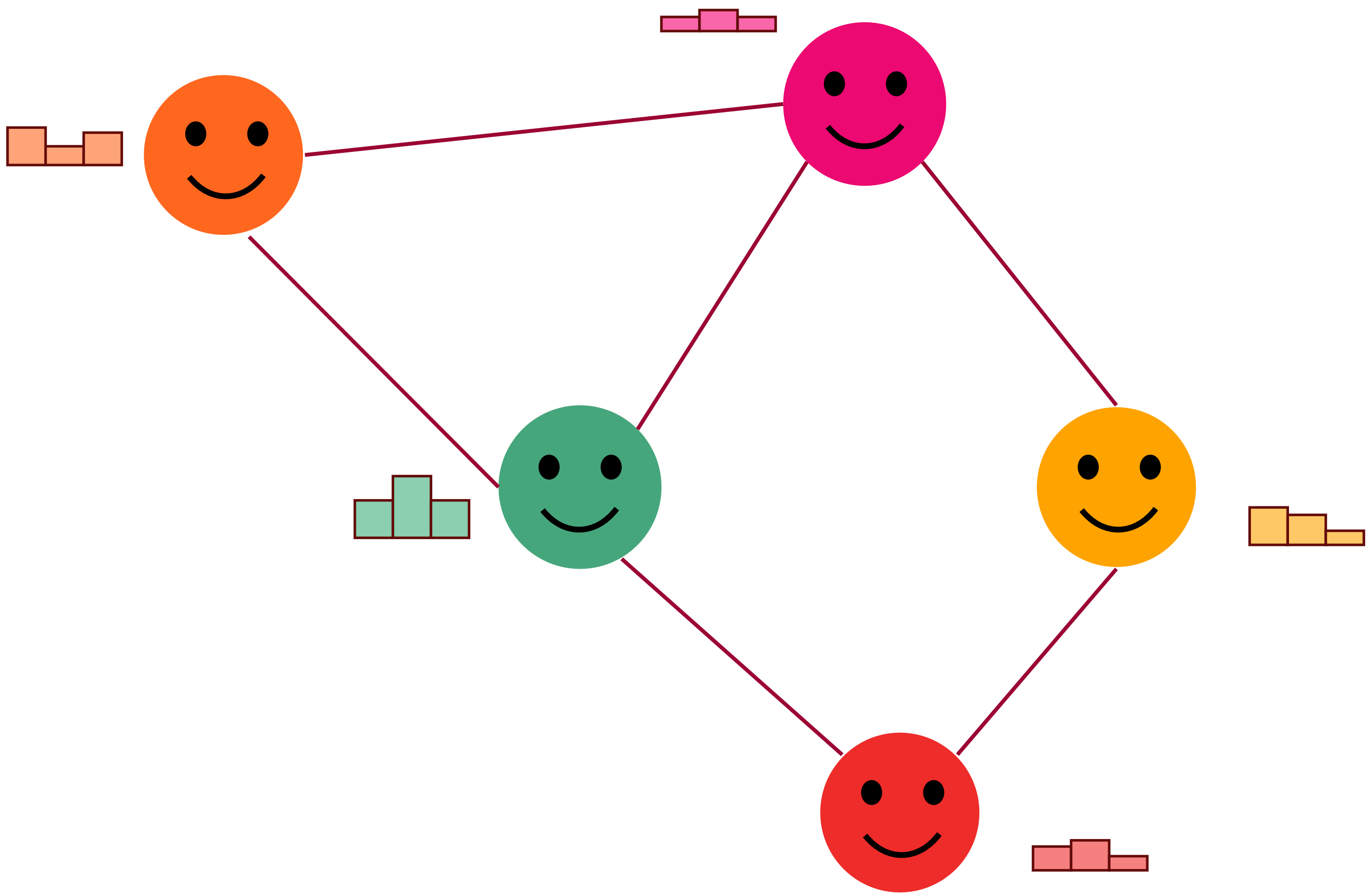


is not the same as



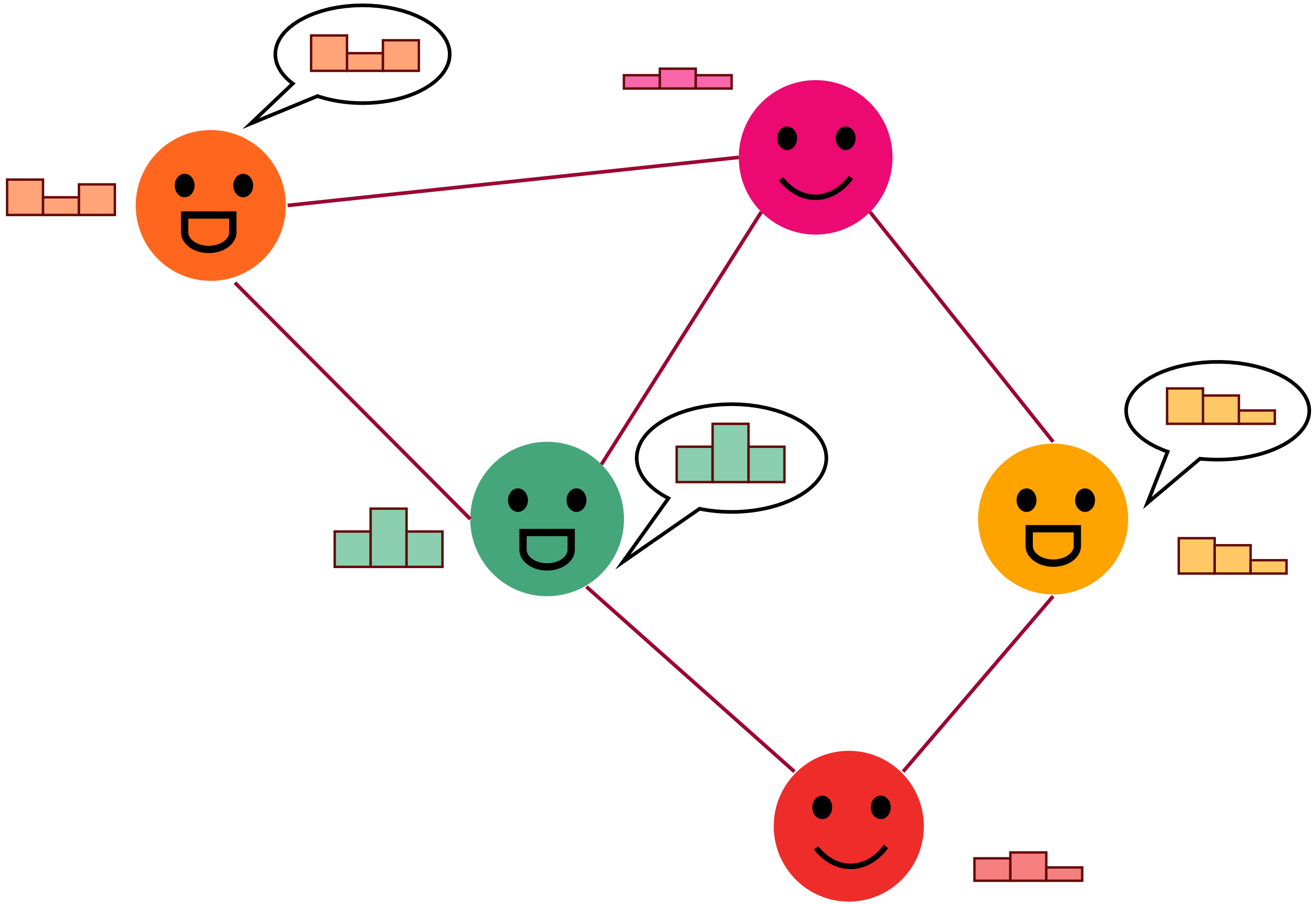
- We want to develop models that can exploit the relationships between the nodes
- We'll introduce **'message passing'** which allows for information to be propagated throughout a neural network

# Message passing



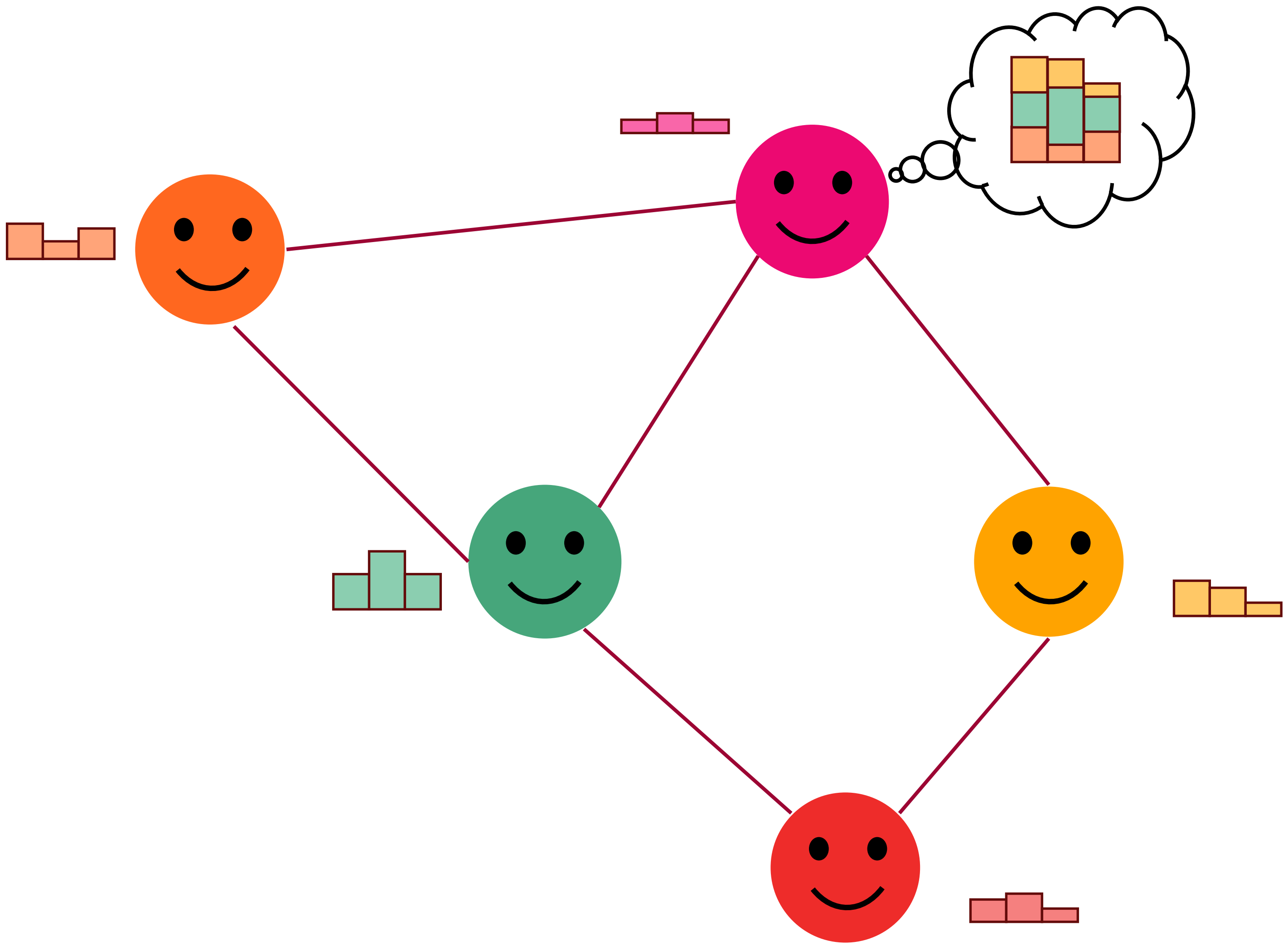
# Message passing

1. **Message:** Nodes send messages to their neighbors



# Message passing

1. **Message:** Nodes send messages to their neighbors
2. **Aggregate:** Messages are aggregated in a permutation-invariant way

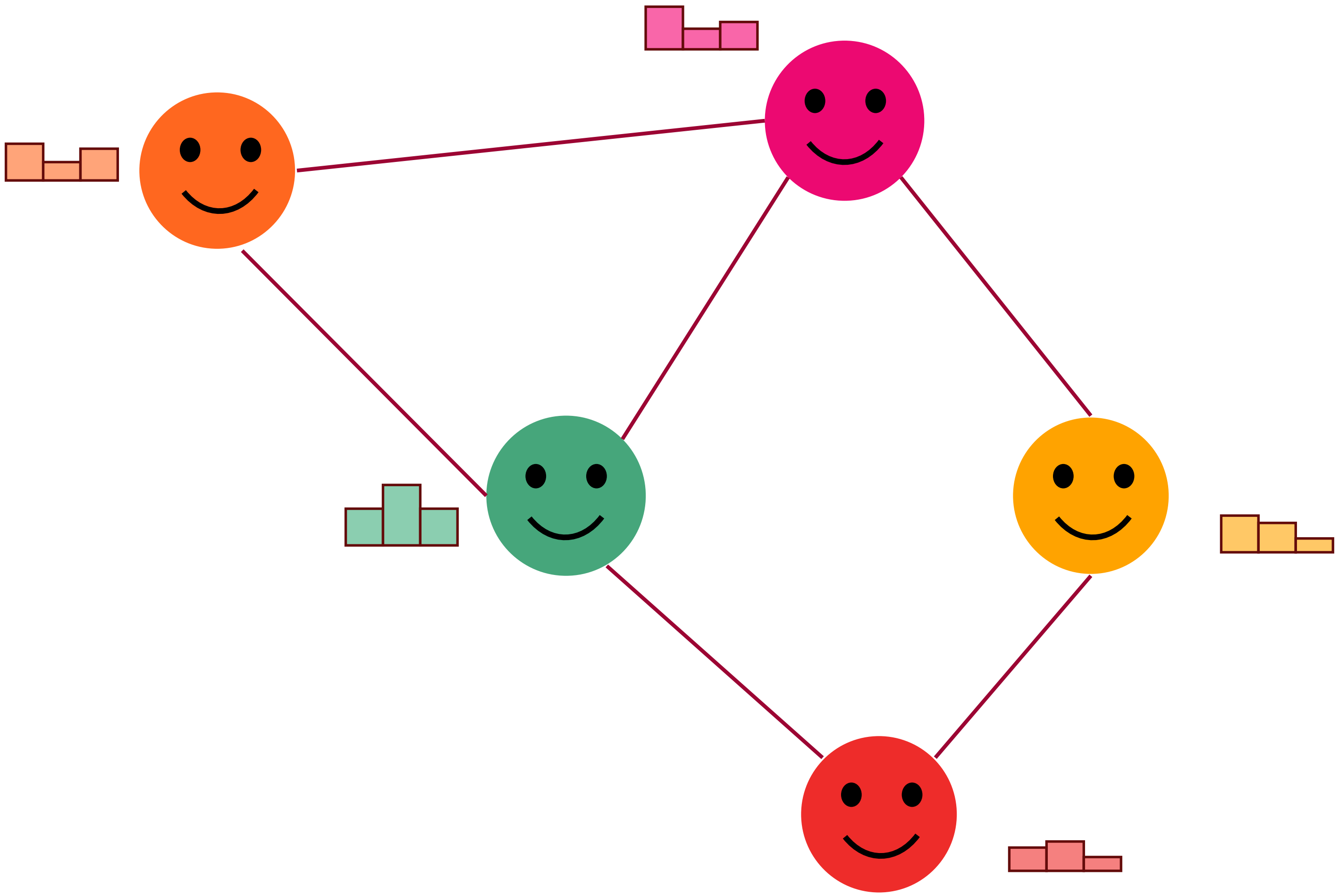




UNIVERSITY OF CALGARY

# Message passing

1. **Message:** Nodes send messages to their neighbors
2. **Aggregate:** Messages are aggregated in a permutation-invariant way
3. **Update:** Node embeddings are updated

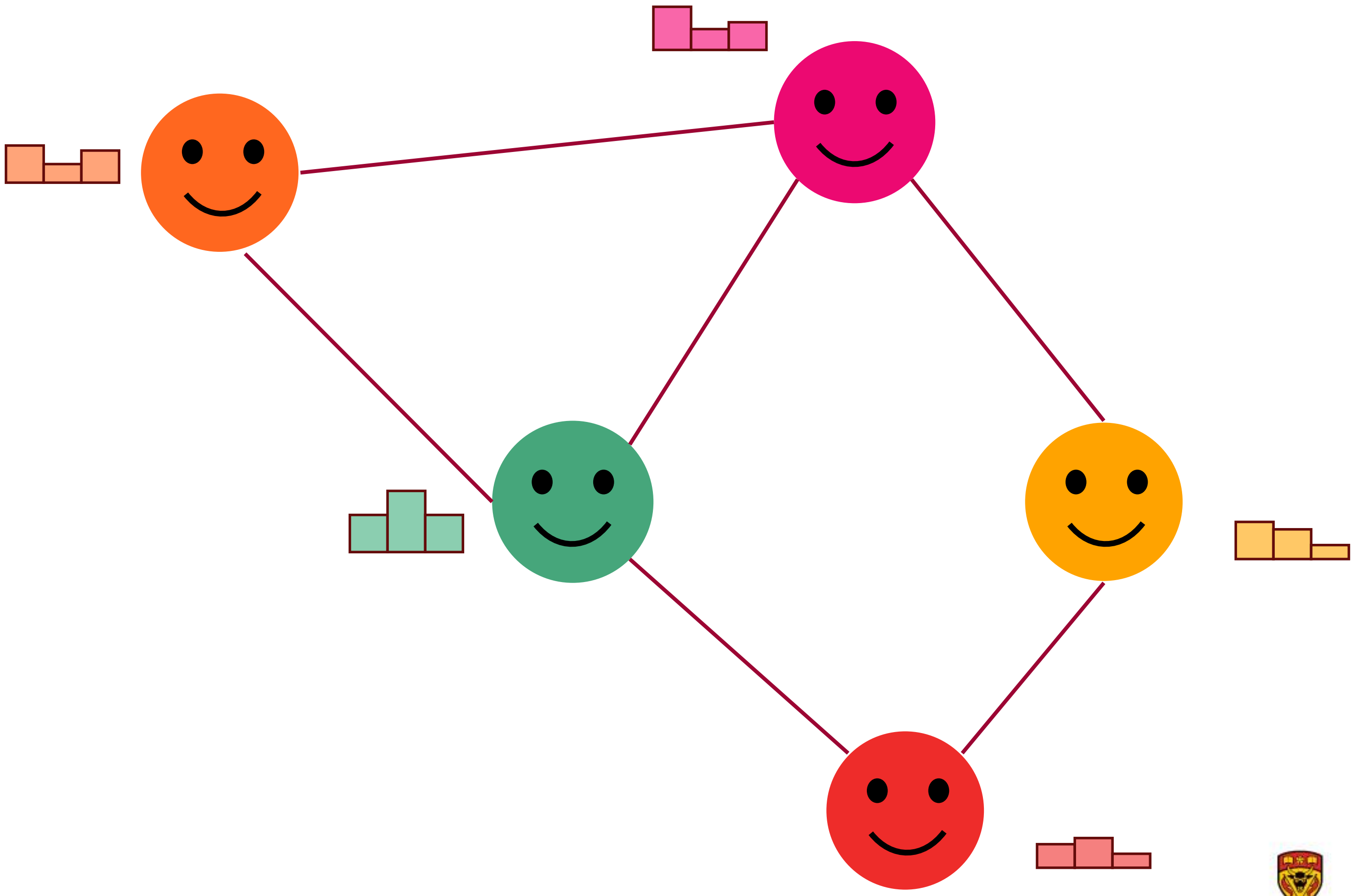




UNIVERSITY OF CALGARY

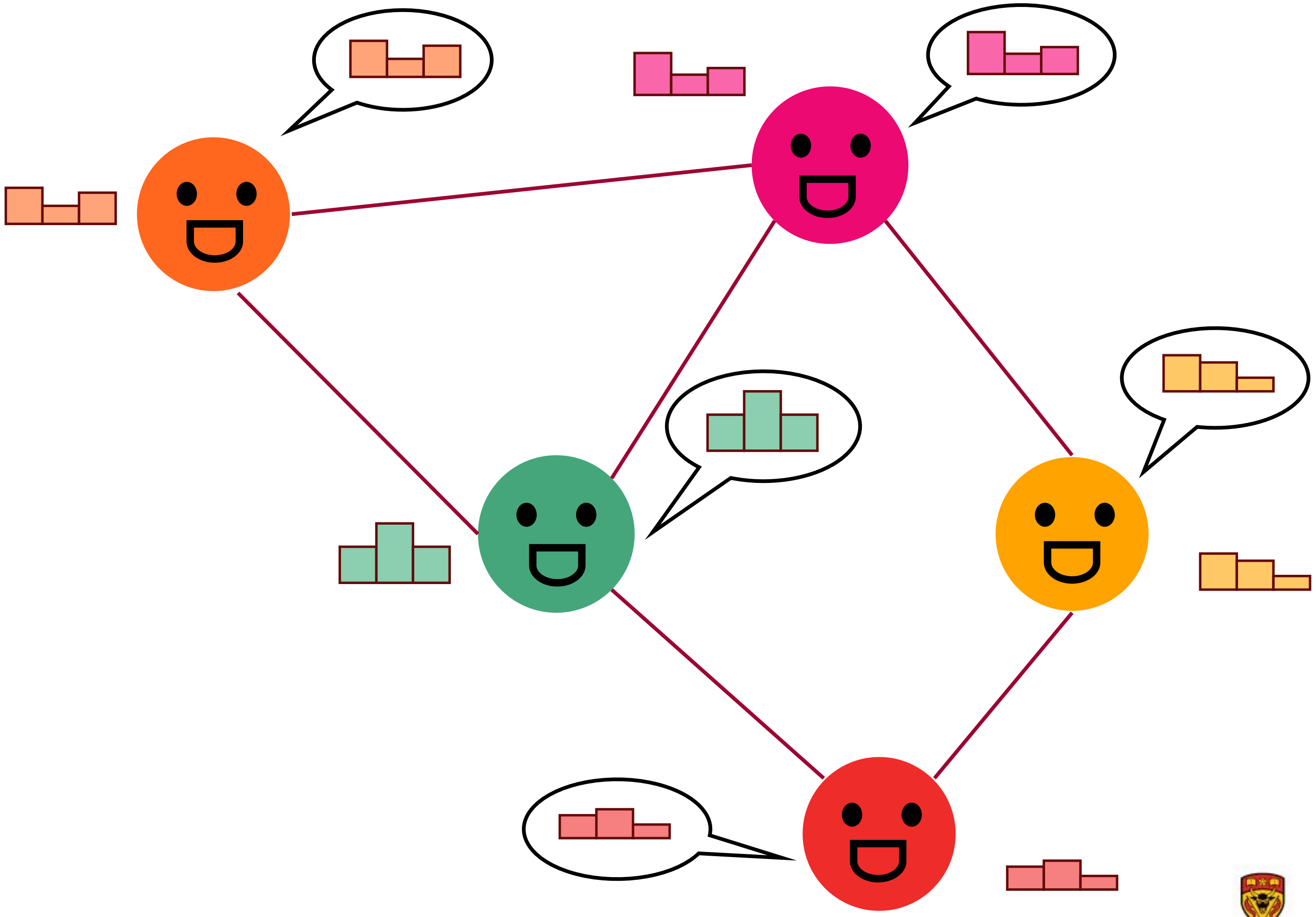
# Message passing

- We just focused on updating the pink node, but in practice, all nodes are updated at the same time!



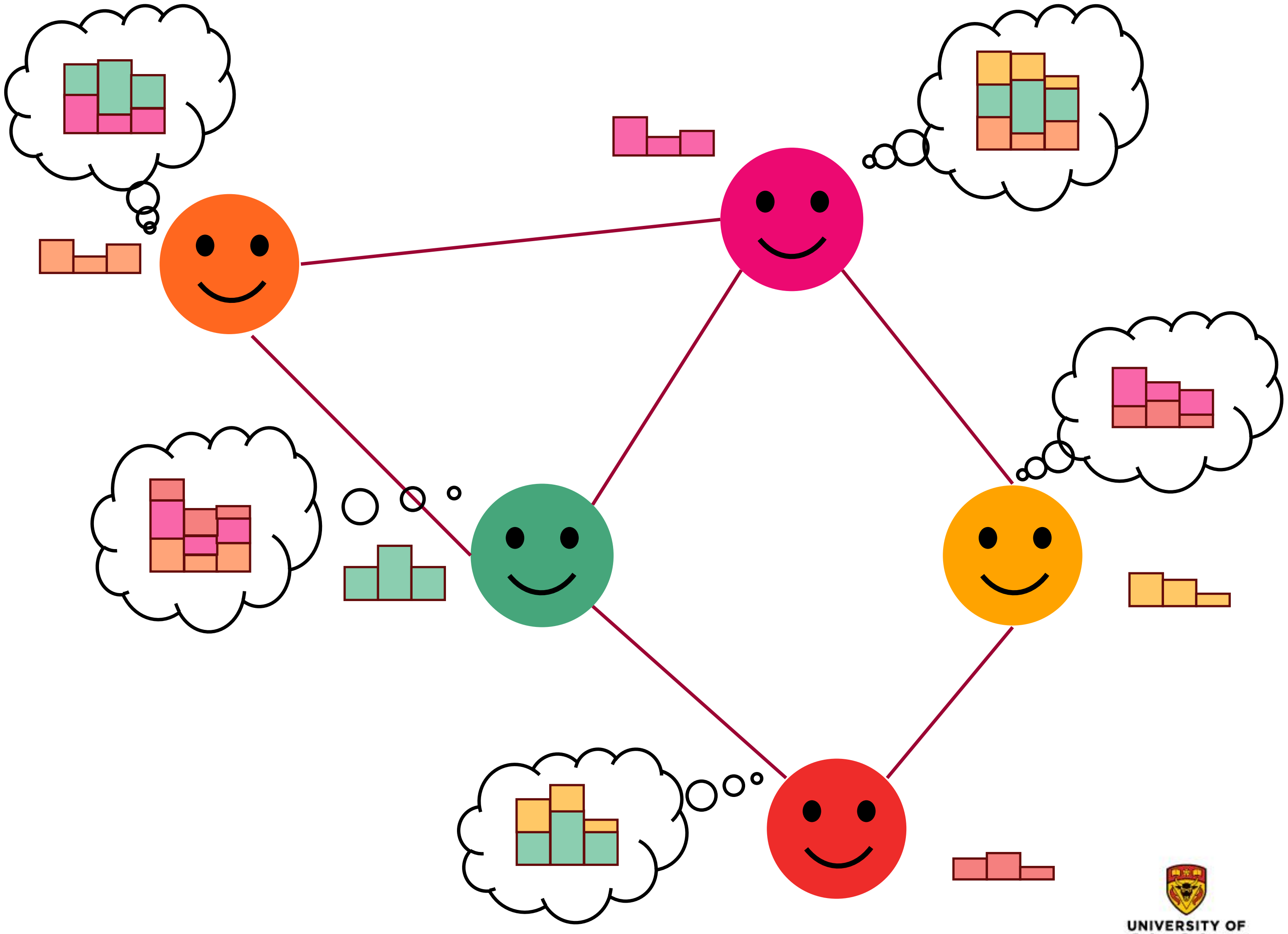
# Message passing

- We just focused on updating the pink node, but in practice, all nodes are updated at the same time!



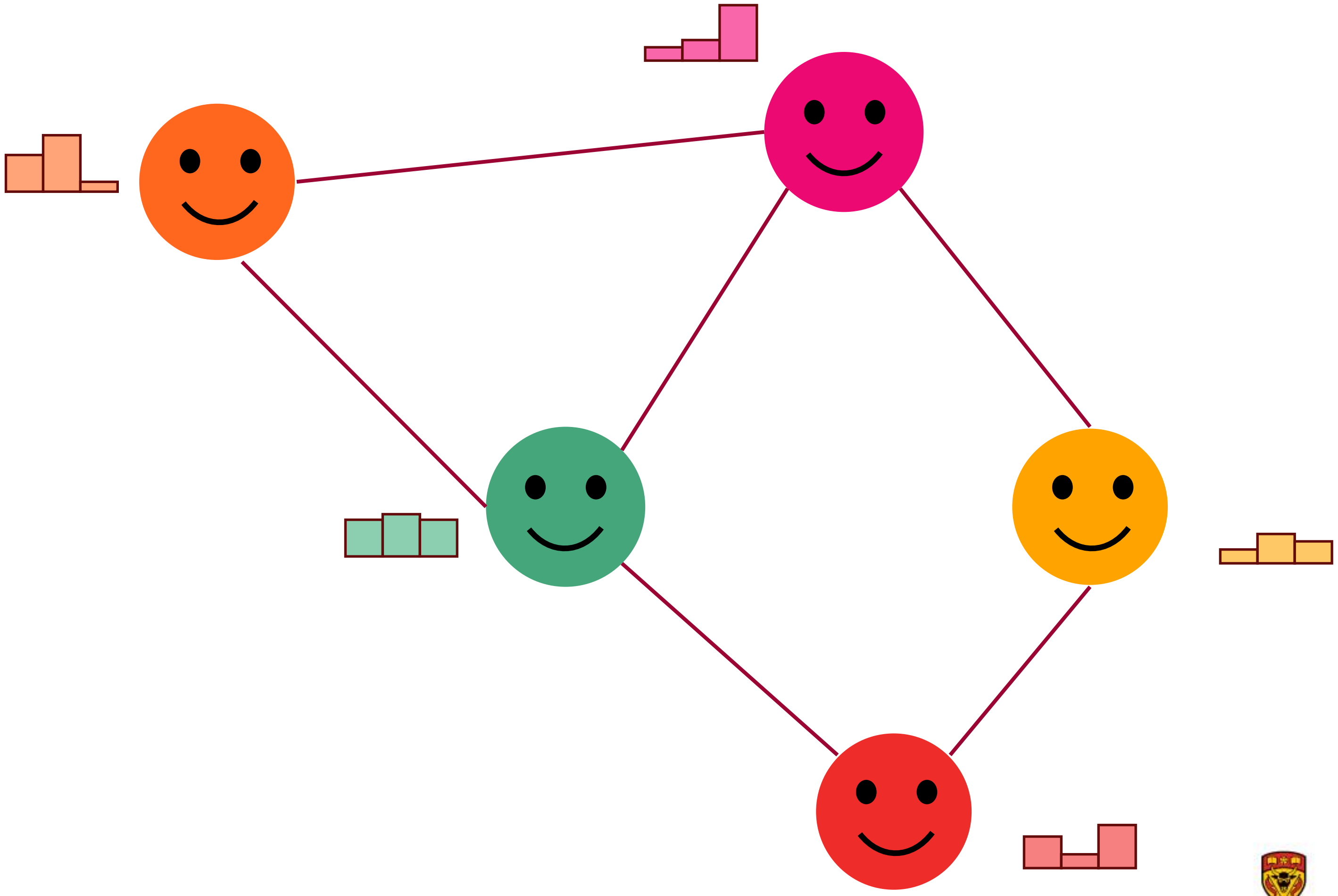
# Message passing

- We just focused on updating the pink node, but in practice, all nodes are updated at the same time!



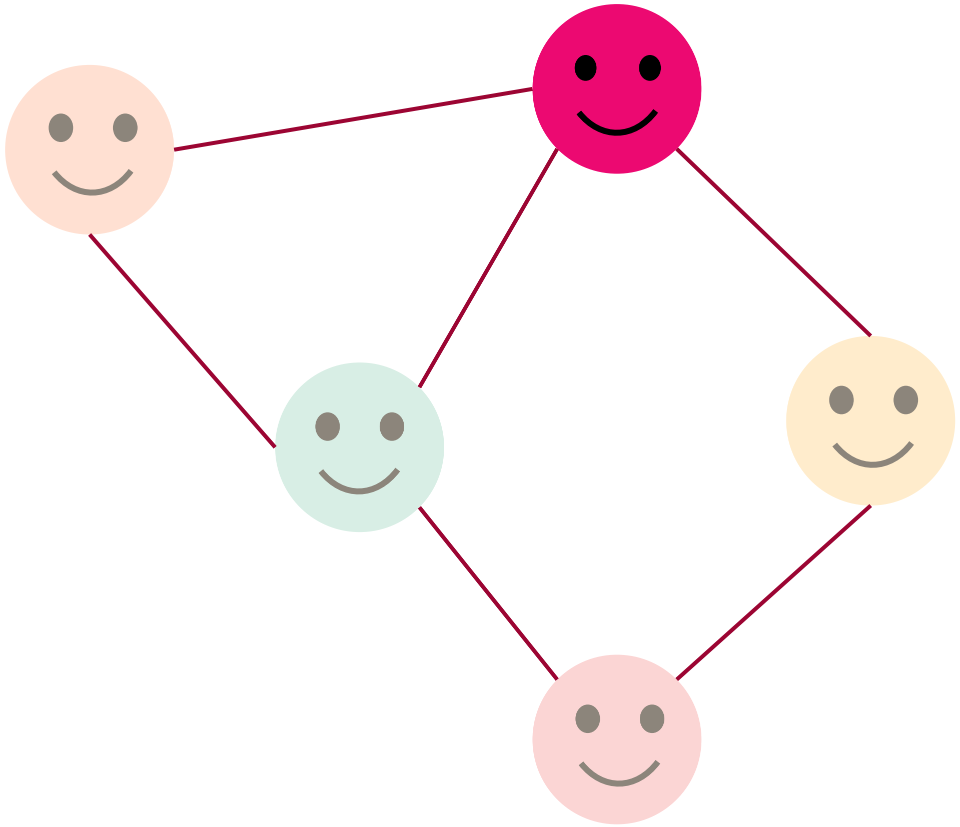
# Message passing

- We just focused on updating the pink node, but in practice, all nodes are updated at the same time!



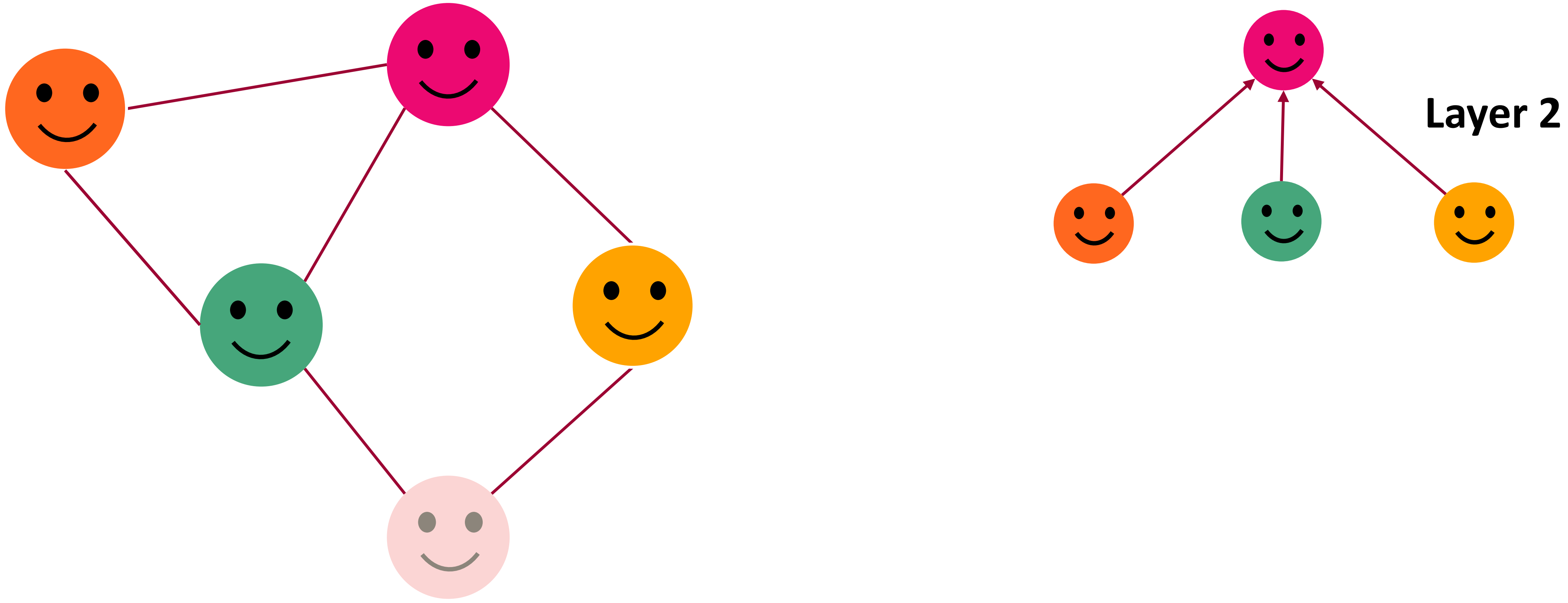
# k-hop neighborhood

- The number of layers we have corresponds to the 'k-hop' neighborhood



# k-hop neighborhood

- The number of layers we have corresponds to the ‘k-hop’ neighborhood

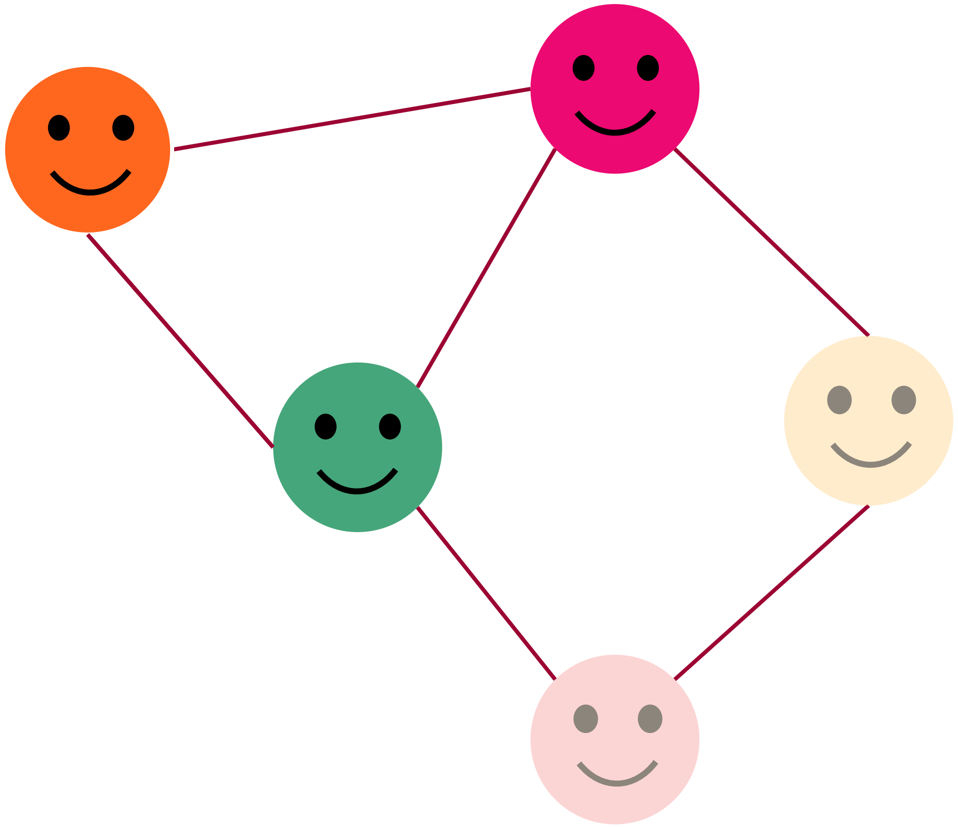


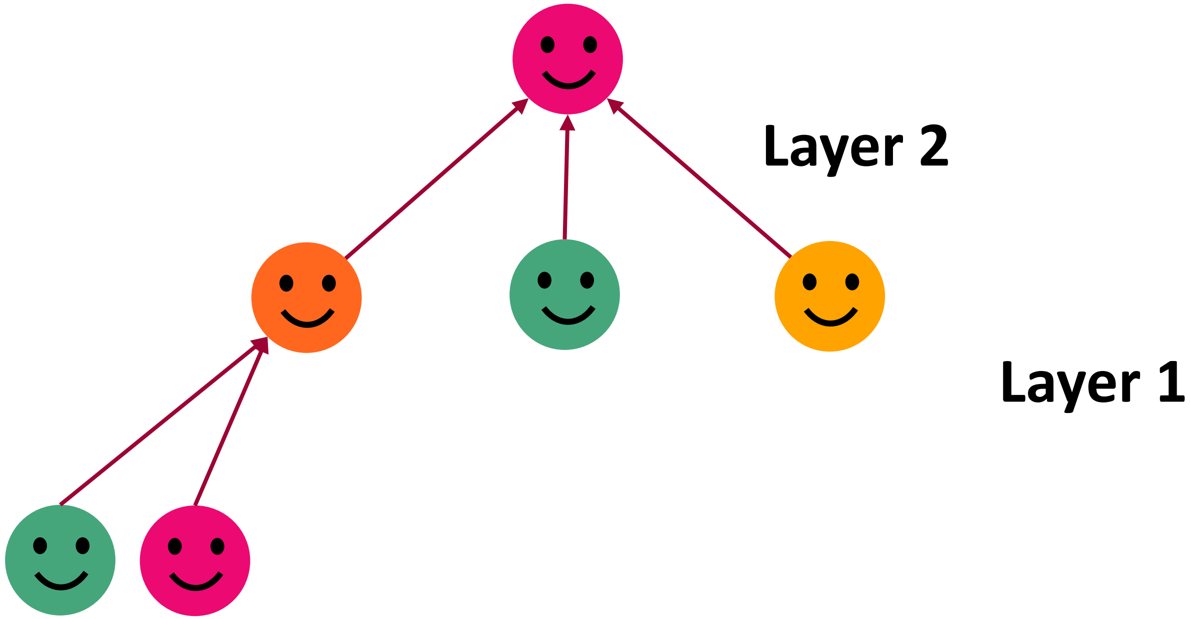



UNIVERSITY OF CALGARY

# k-hop neighborhood

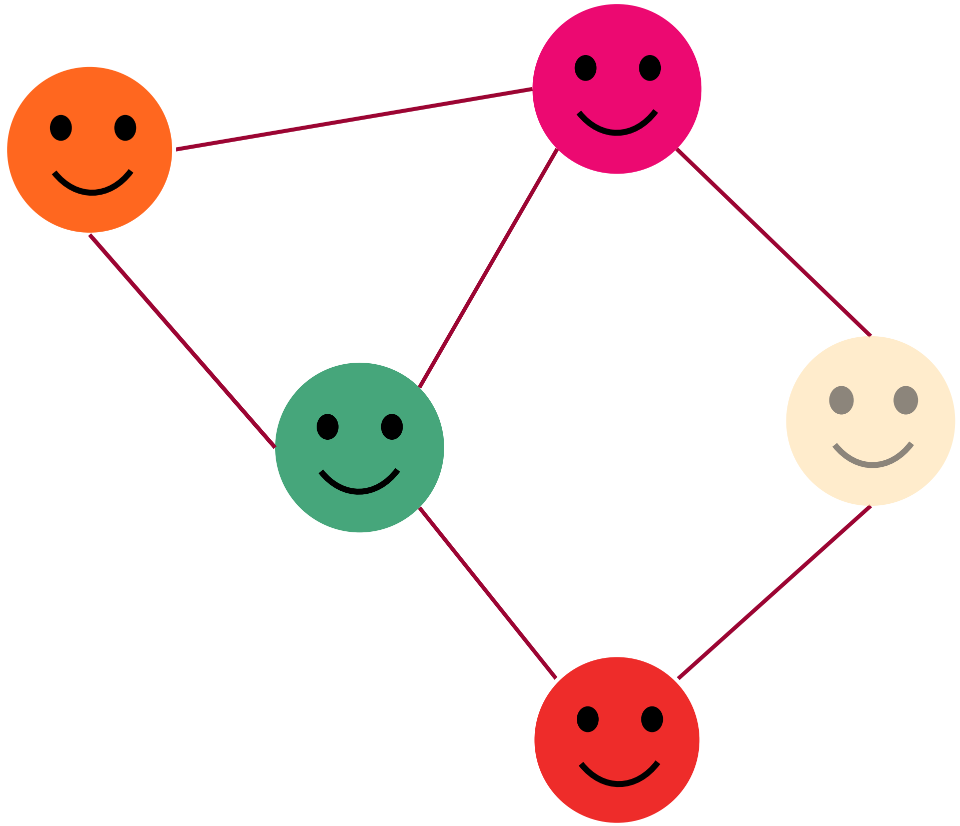
- The number of layers we have corresponds to the ‘k-hop’ neighborhood

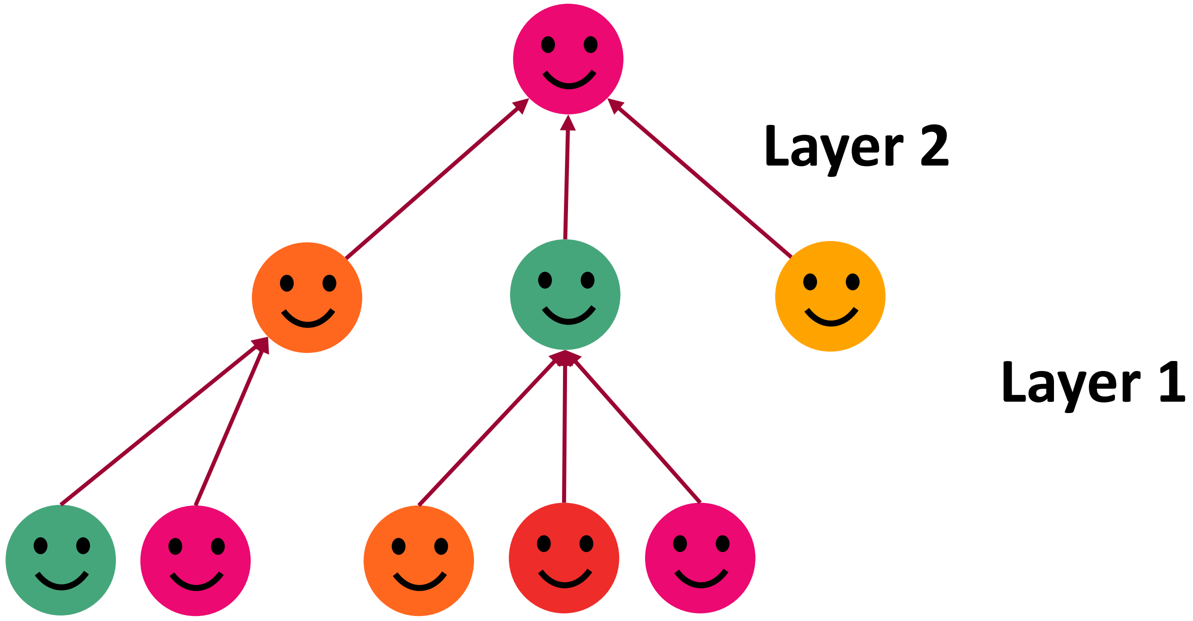




# k-hop neighborhood

- The number of layers we have corresponds to the 'k-hop' neighborhood



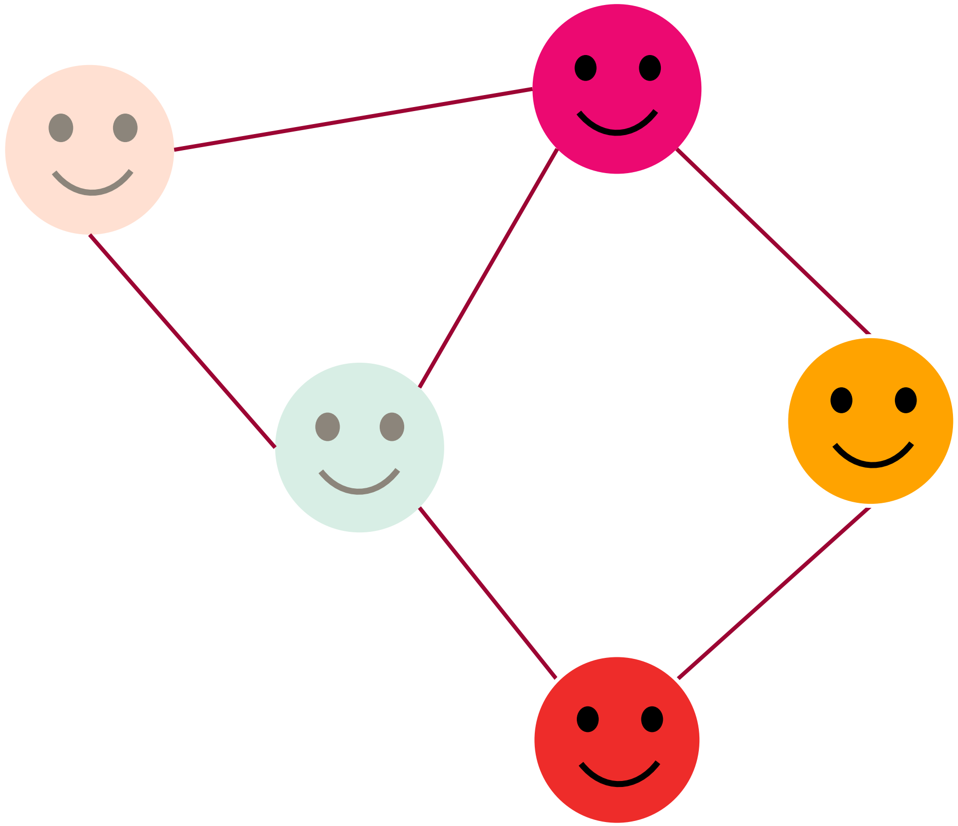


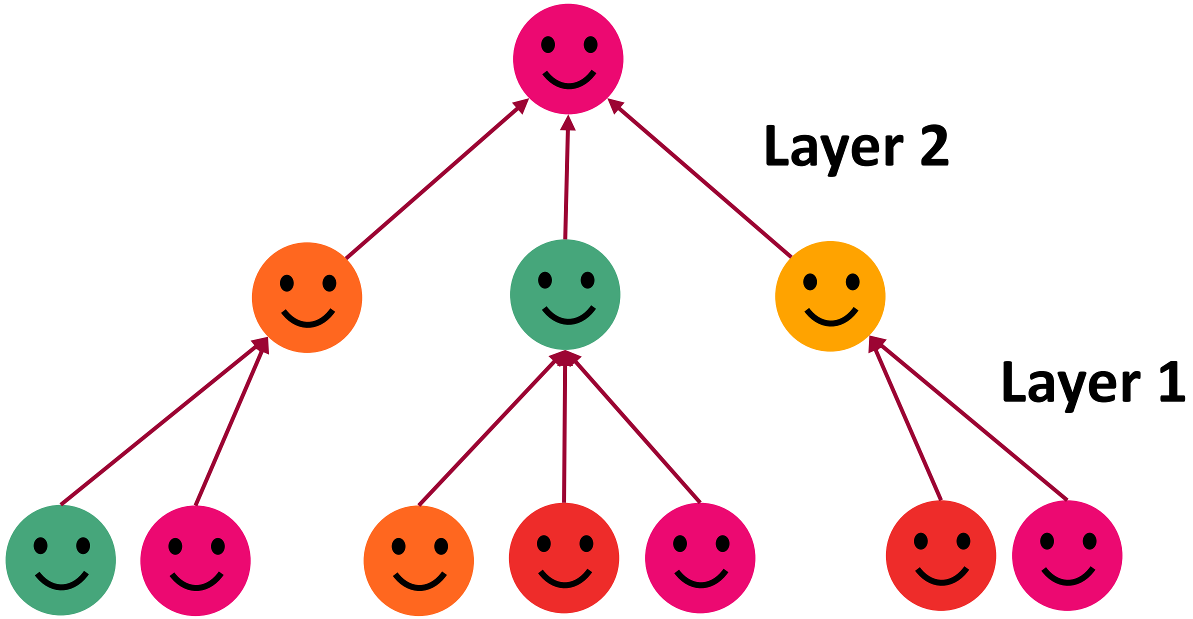



UNIVERSITY OF CALGARY

# k-hop neighborhood

- The number of layers we have corresponds to the ‘k-hop’ neighborhood



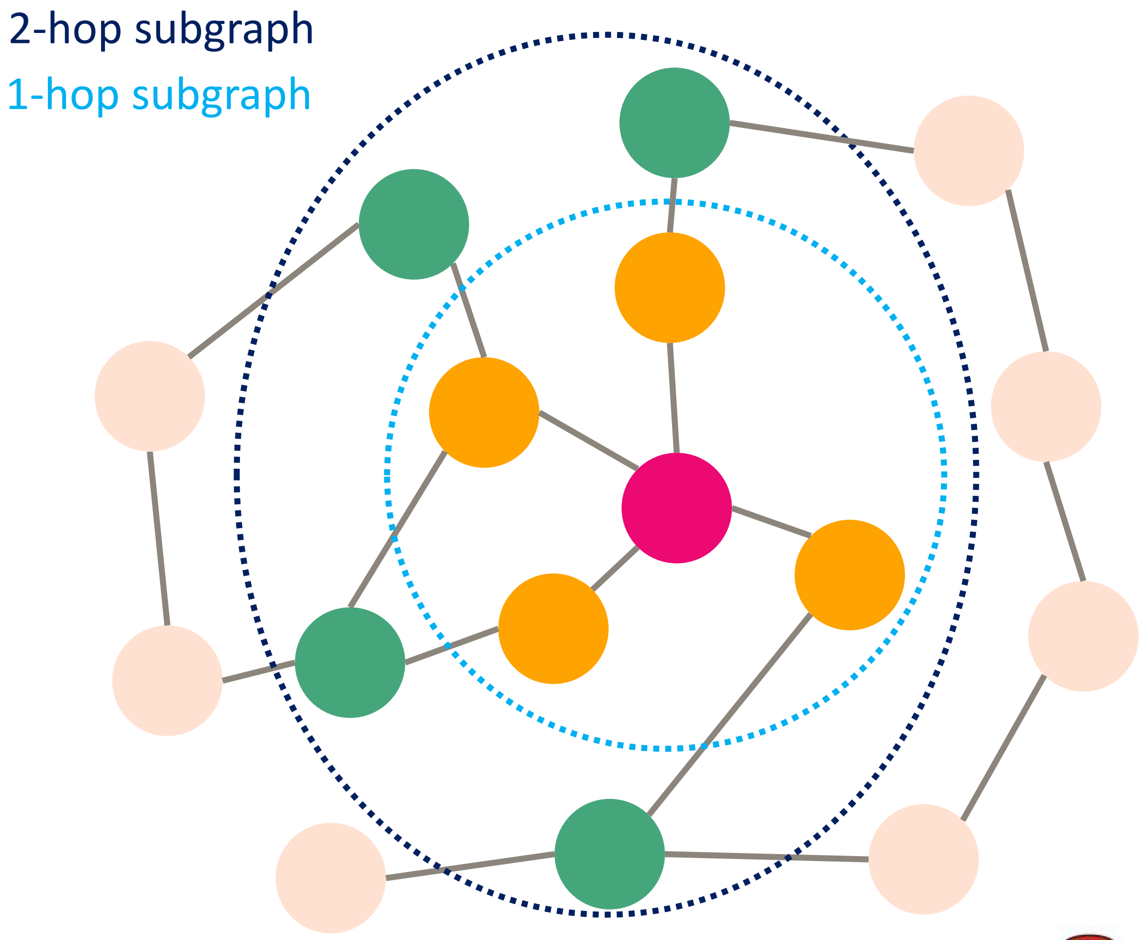





UNIVERSITY OF CALGARY

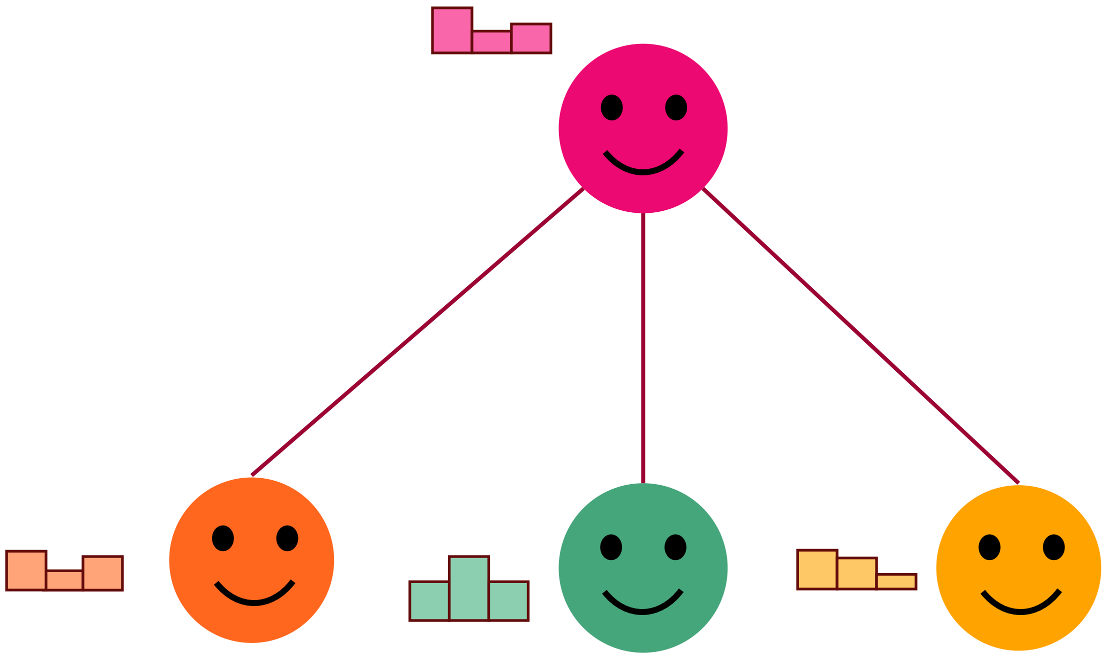
# k-hop neighborhood

- The more hops, the more distant information each node gets
- More hops is good for global context
- …but too many hops will wash out local structural information



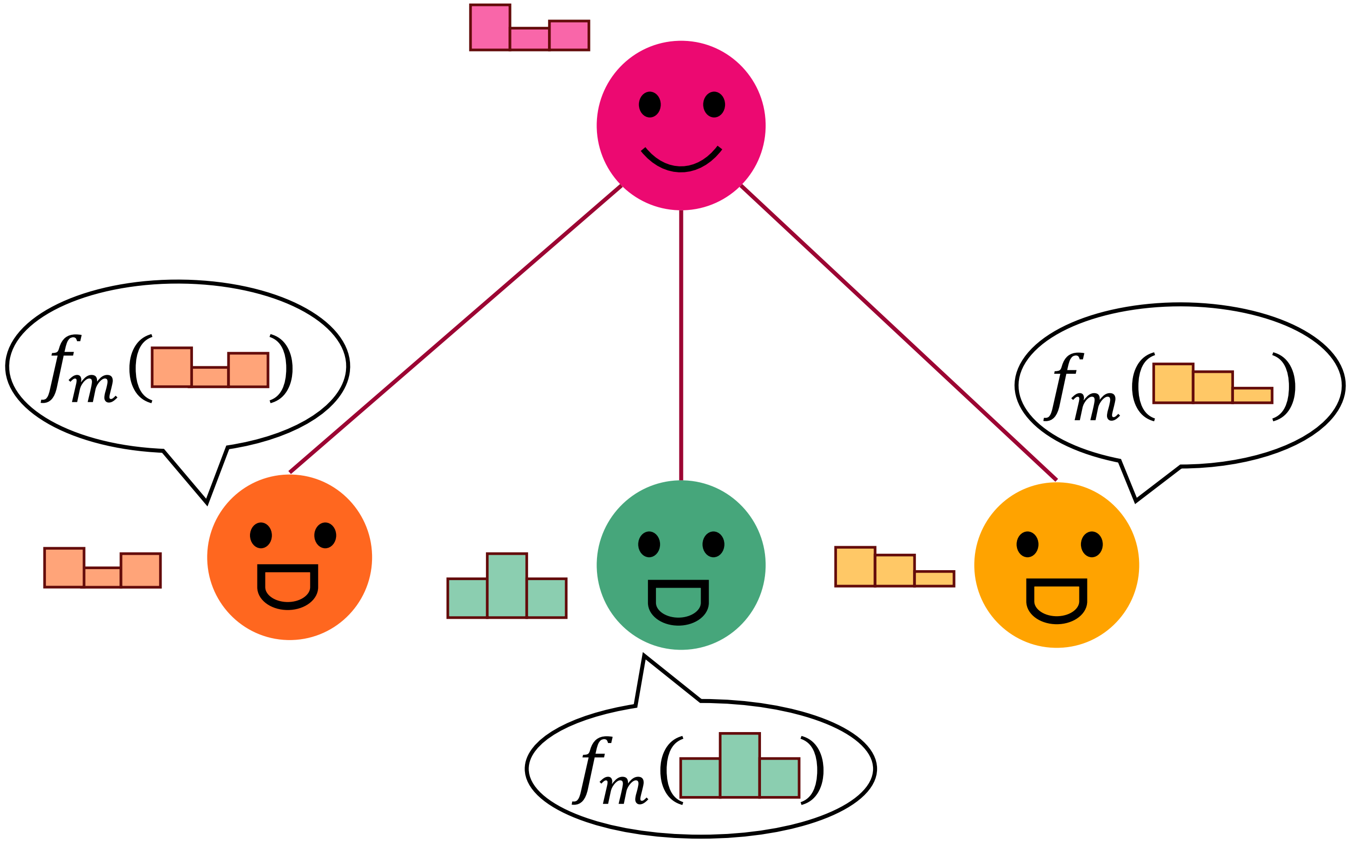
# Trainable parameters

- Can think of it as two places we can have trainable parameters:



# Trainable parameters

- Can think of it as two places we can have trainable parameters:
  - When passing a message

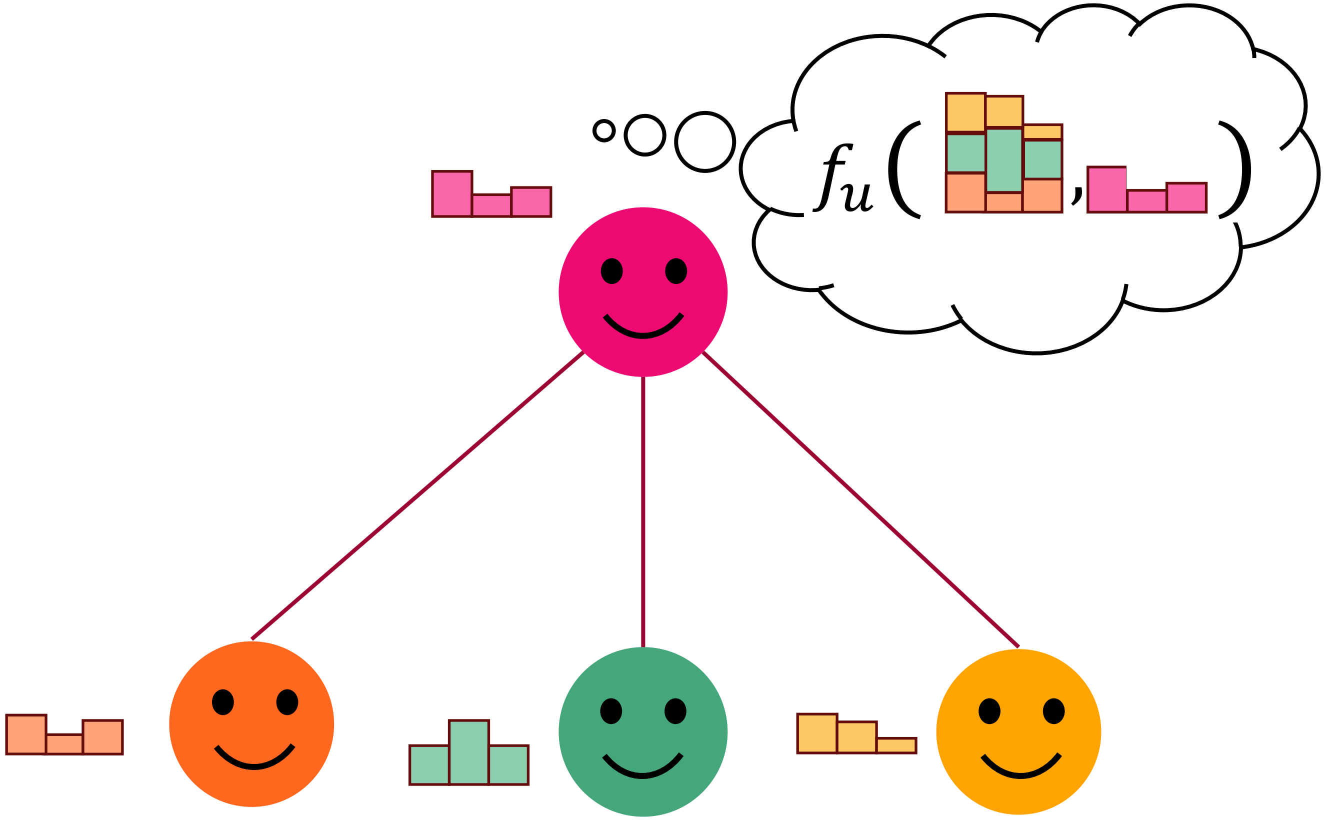





UNIVERSITY OF CALGARY

# Trainable parameters

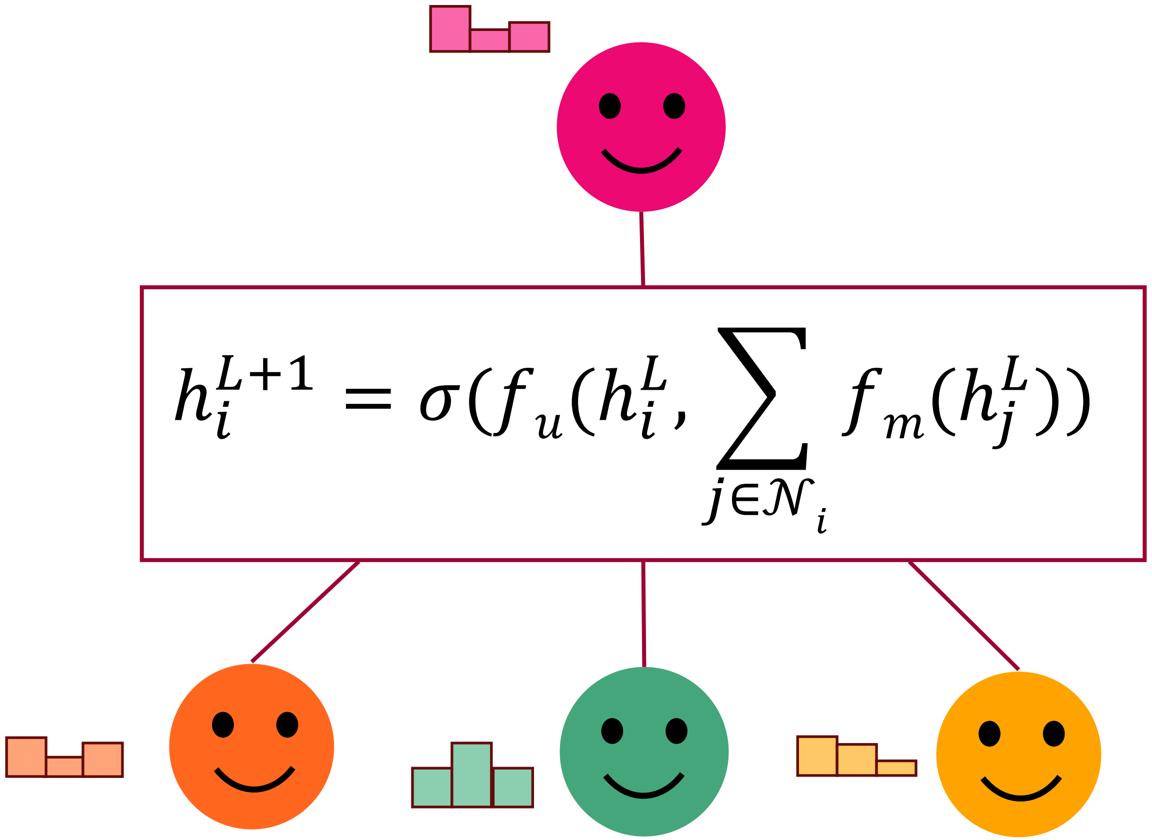
- Can think of it as two places we can have trainable parameters:
  - When passing a message
  - When updating an embedding



# Trainable parameters

- Can think of it as two places we can have trainable parameters:
  - When passing a message
  - When updating an embedding

* Note: The trainable parameters are shared across ALL nodes!



$$h_i^{L+1} = \sigma(f_u(h_i^L, \sum_{j \in \mathcal{N}_i} f_m(h_j^L)))$$

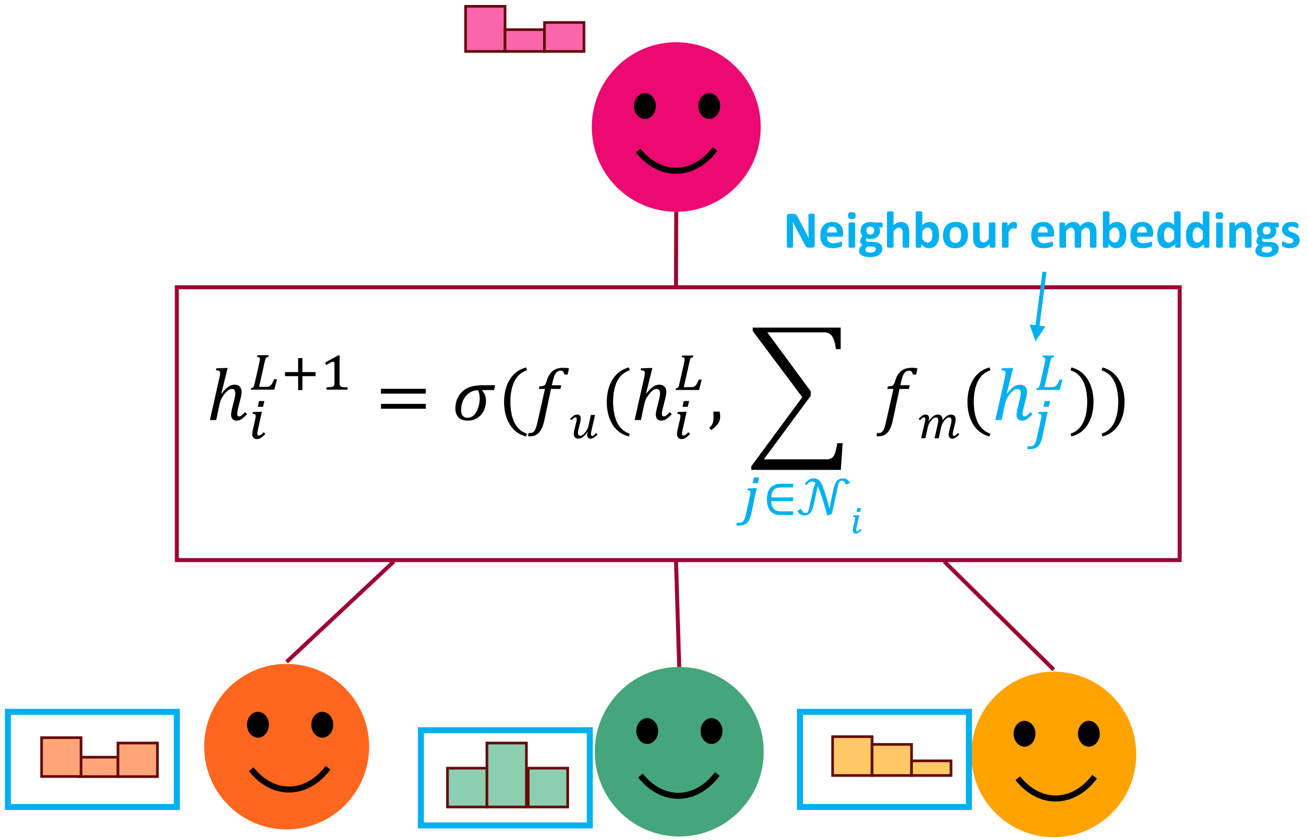


UNIVERSITY OF CALGARY

# Trainable parameters

- Can think of it as two places we can have trainable parameters:
  - When passing a message
  - When updating an embedding

* Note: The trainable parameters are shared across ALL nodes!

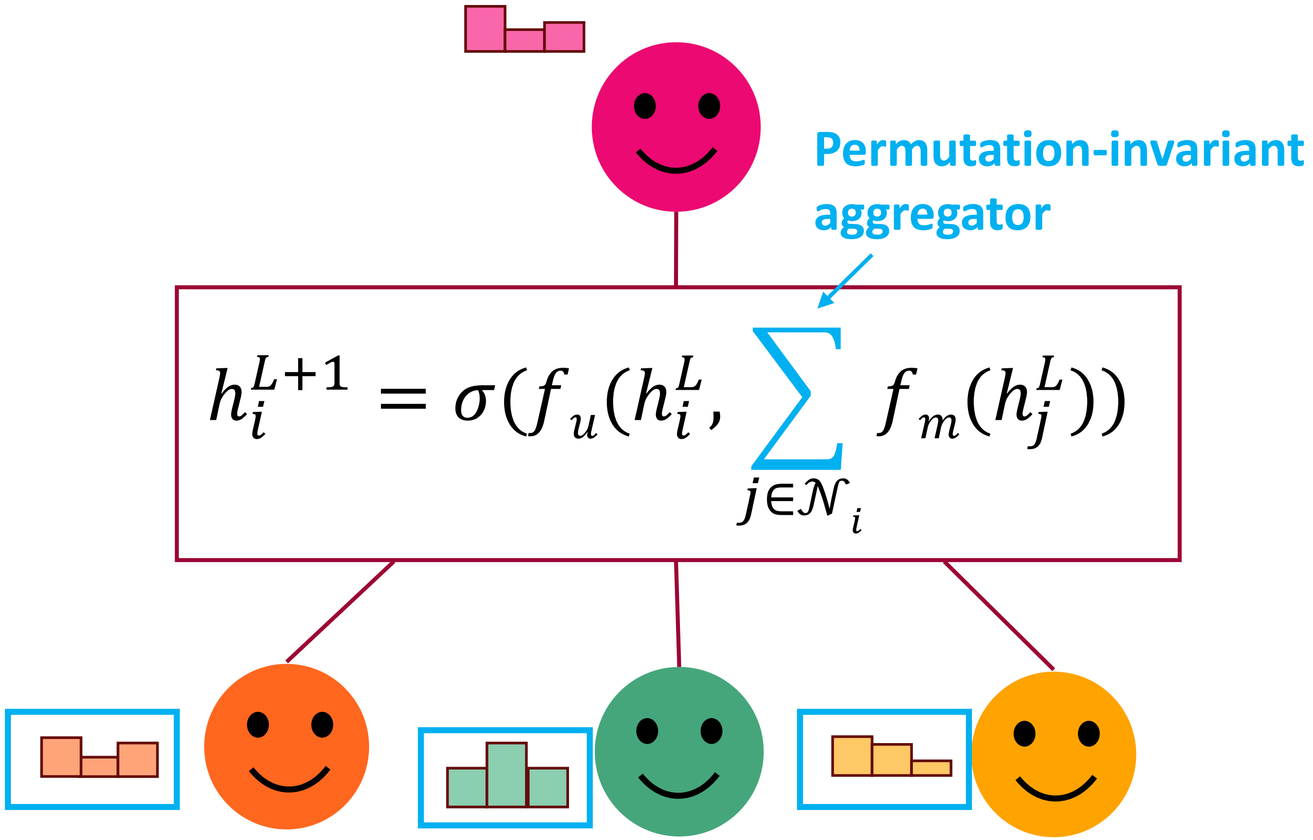



$$h_i^{L+1} = \sigma\left(f_u\left(h_i^L, \sum_{j \in \mathcal{N}_i} f_m\left(h_j^L\right)\right)\right)$$

# Trainable parameters

- Can think of it as two places we can have trainable parameters:
  - When passing a message
  - When updating an embedding

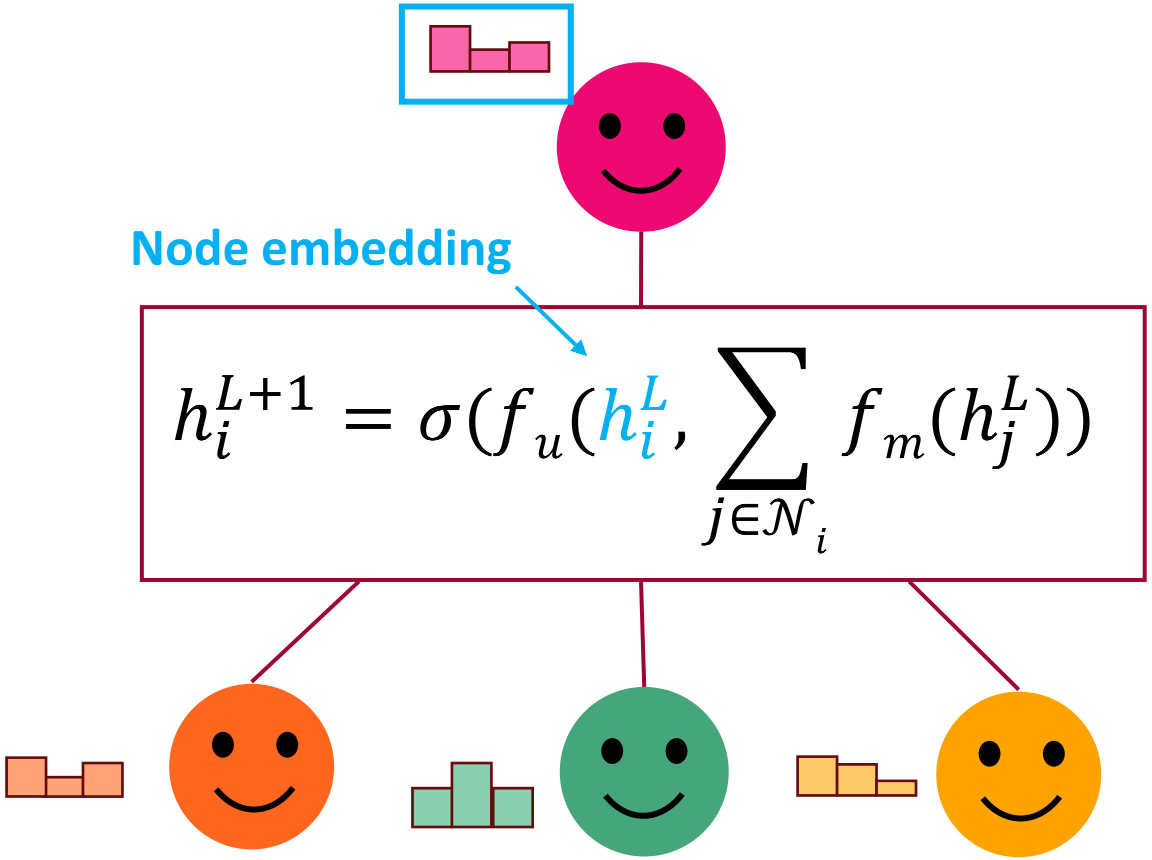
* Note: The trainable parameters are shared across ALL nodes!



# Trainable parameters

- Can think of it as two places we can have trainable parameters:
  - When passing a message
  - When updating an embedding

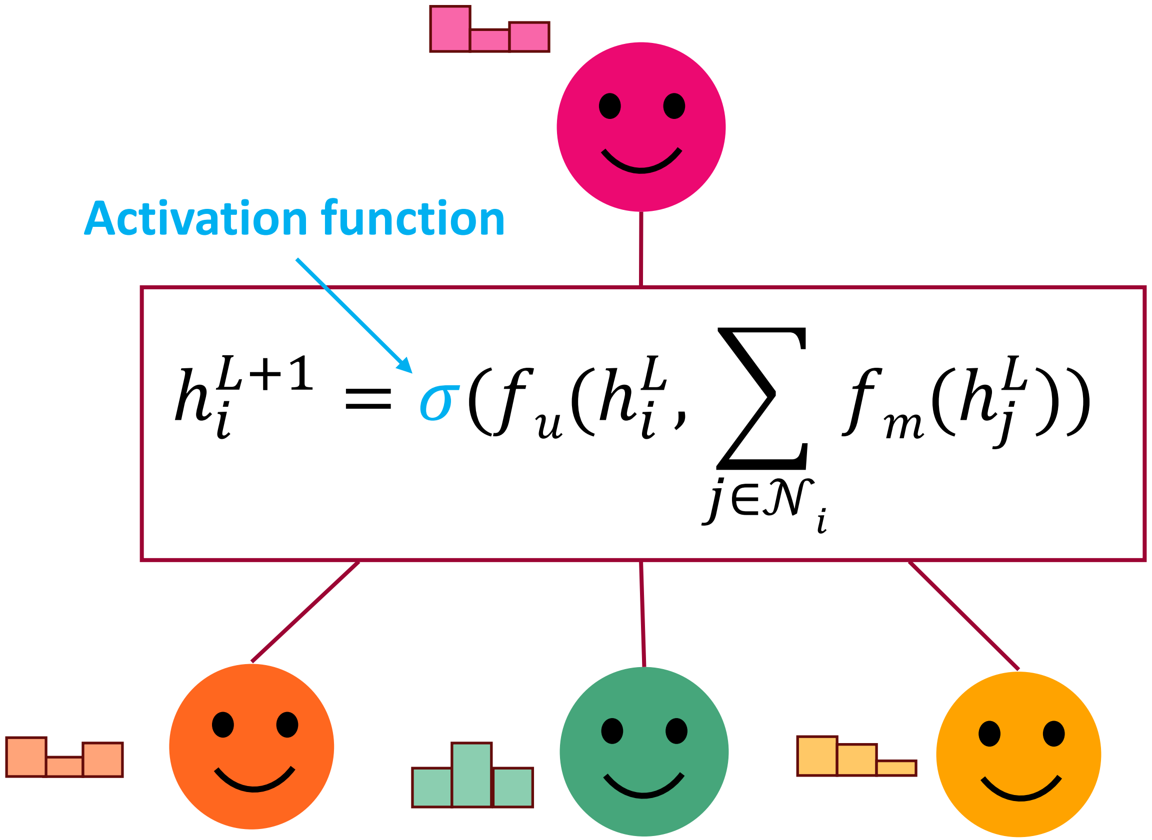
* Note: The trainable parameters are shared across ALL nodes!




$$h_i^{L+1} = \sigma(f_u(h_i^L, \sum_{j \in \mathcal{N}_i} f_m(h_j^L)))$$

# Trainable parameters

- Can think of it as two places we can have trainable parameters:
  - When passing a message
  - When updating an embedding

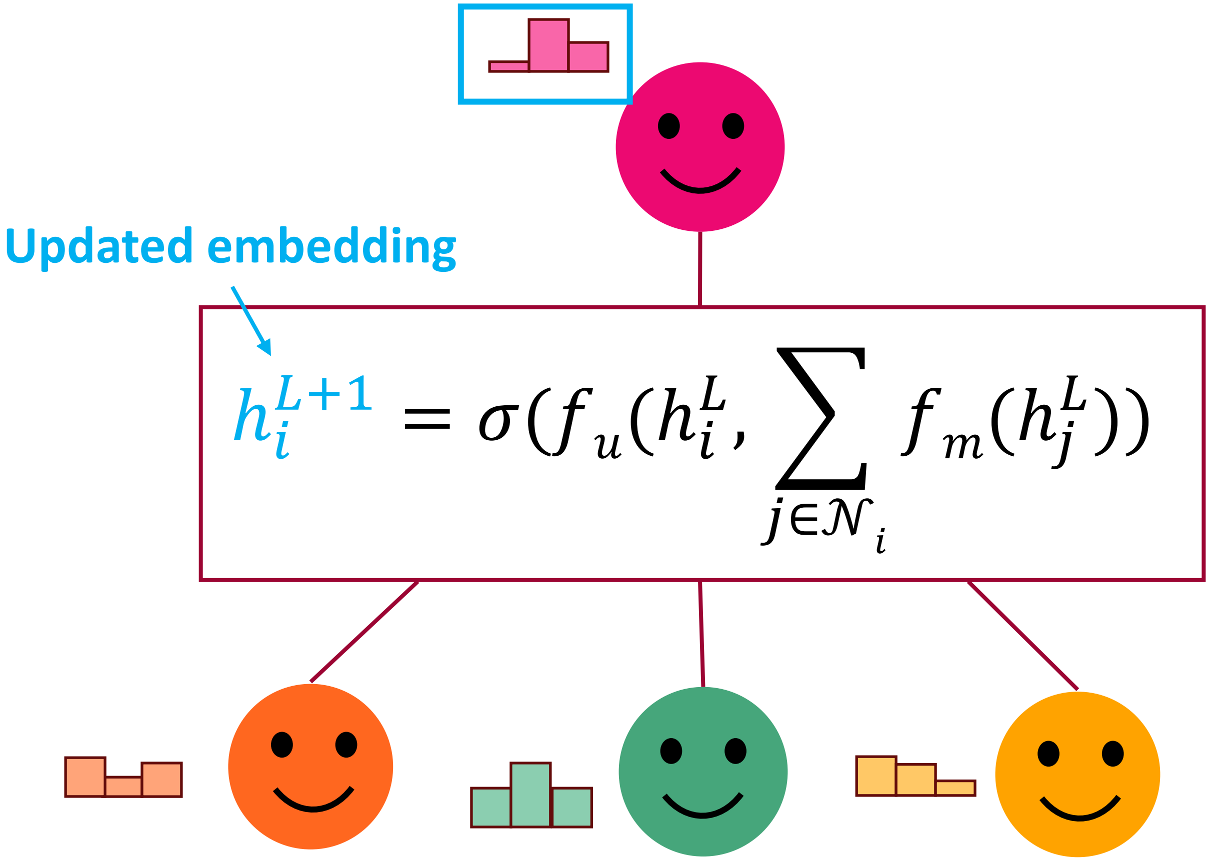



* Note: The trainable parameters are shared across ALL nodes!

# Trainable parameters

- Can think of it as two places we can have trainable parameters:
  - When passing a message
  - When updating an embedding

* Note: The trainable parameters are shared across ALL nodes!



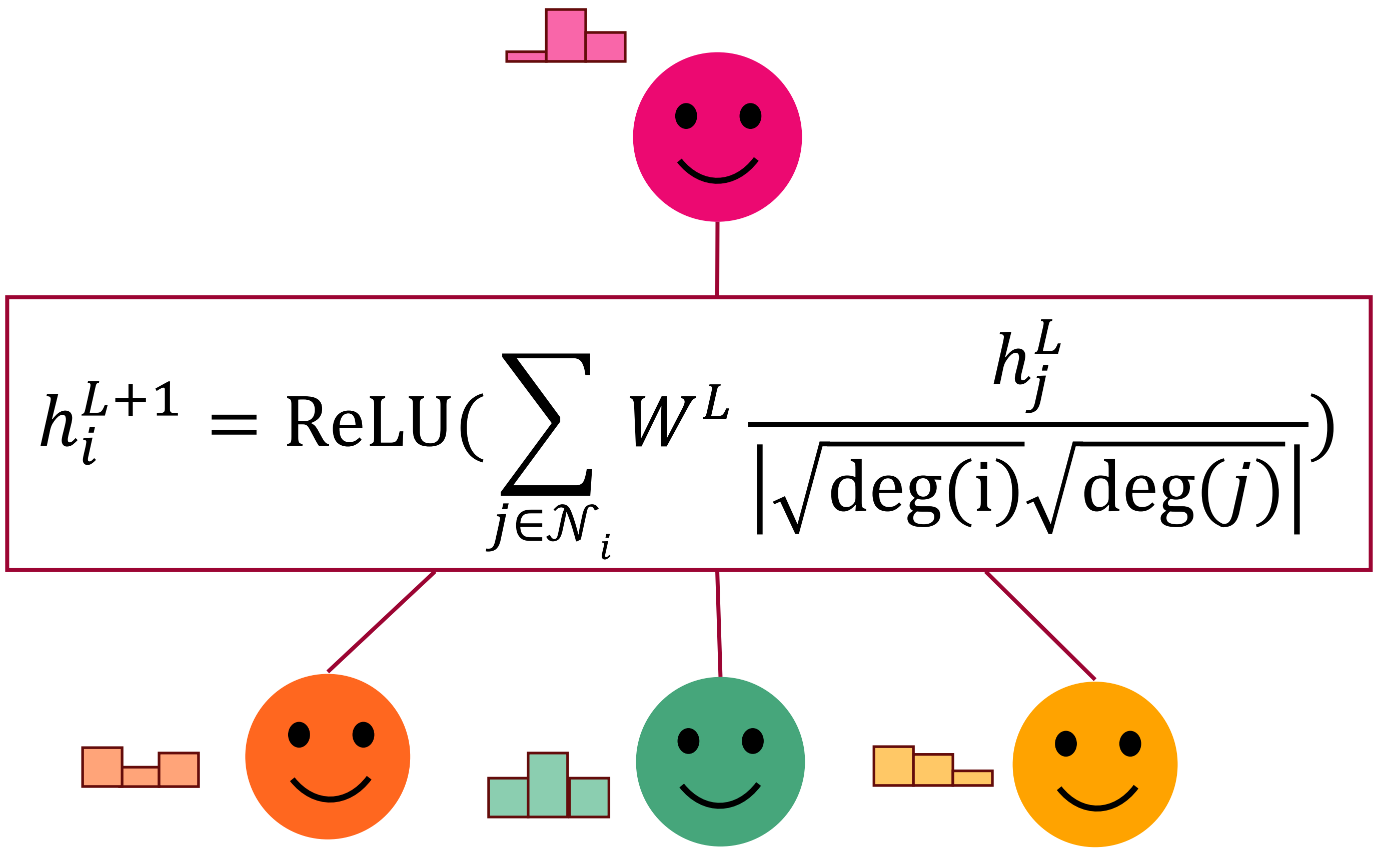

$$h_i^{L+1} = \sigma\left(f_u\left(h_i^L, \sum_{j \in \mathcal{N}_i} f_m\left(h_j^L\right)\right)\right)$$



UNIVERSITY OF CALGARY

# GNN layer example: Graph Convolutional Network

- Graph convolutional network (GCN) learns a simple matrix of weights
  - Can think of this matrix as a ‘convolutional kernel’ we use at each node instead of pixel



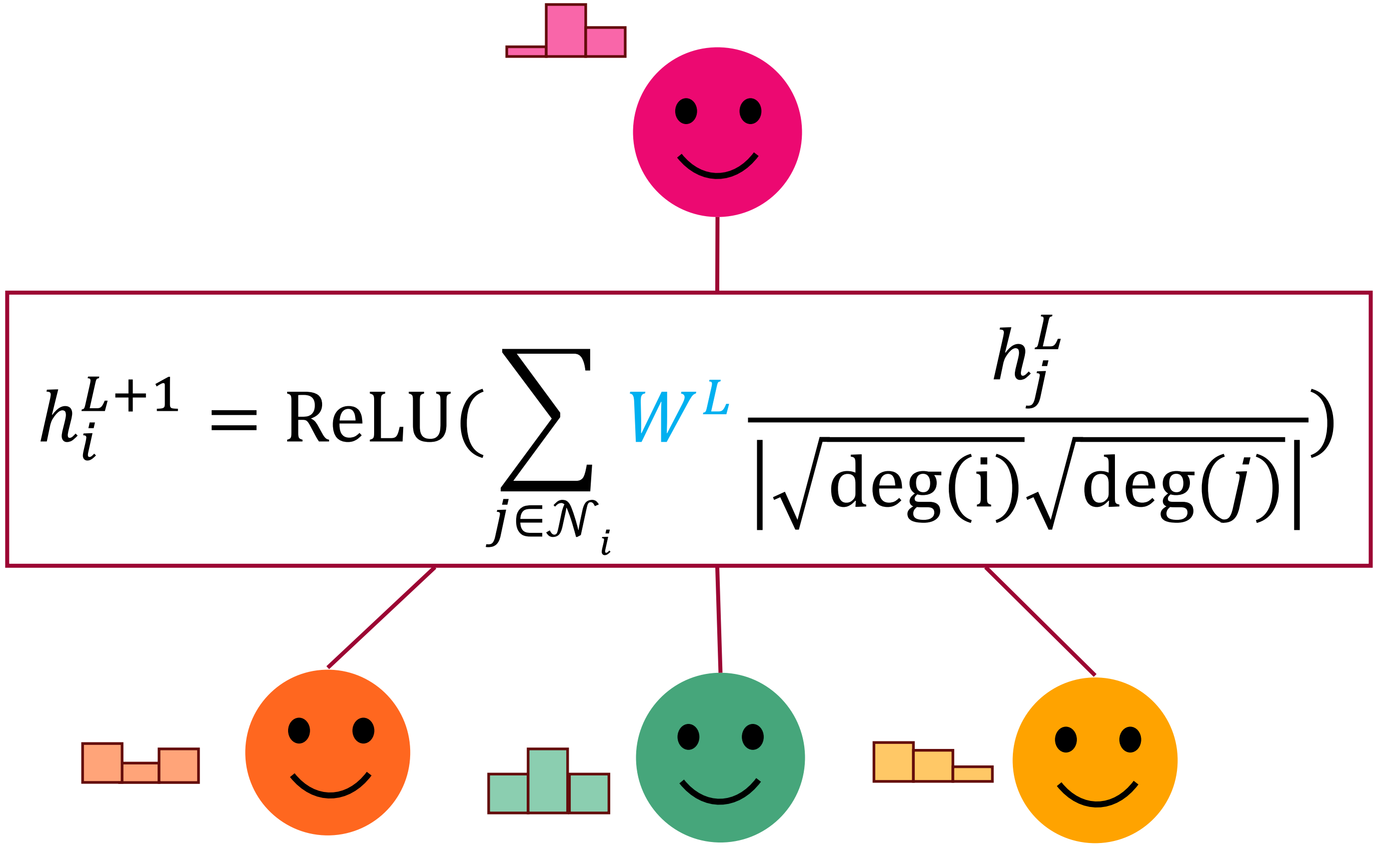
$$h_i^{L+1} = \mathrm{ReLU}\Big(\sum_{j \in \mathcal{N}_i} W^L \frac{h_j^L}{\left|\sqrt{\deg(i)}\sqrt{\deg(j)}\right|}\Big)$$



UNIVERSITY OF CALGARY

# GNN layer example: Graph Convolutional Network

- Graph convolutional network (GCN) learns a simple matrix of weights
  - Can think of this matrix as a ‘convolutional kernel’ we use at each node instead of pixel



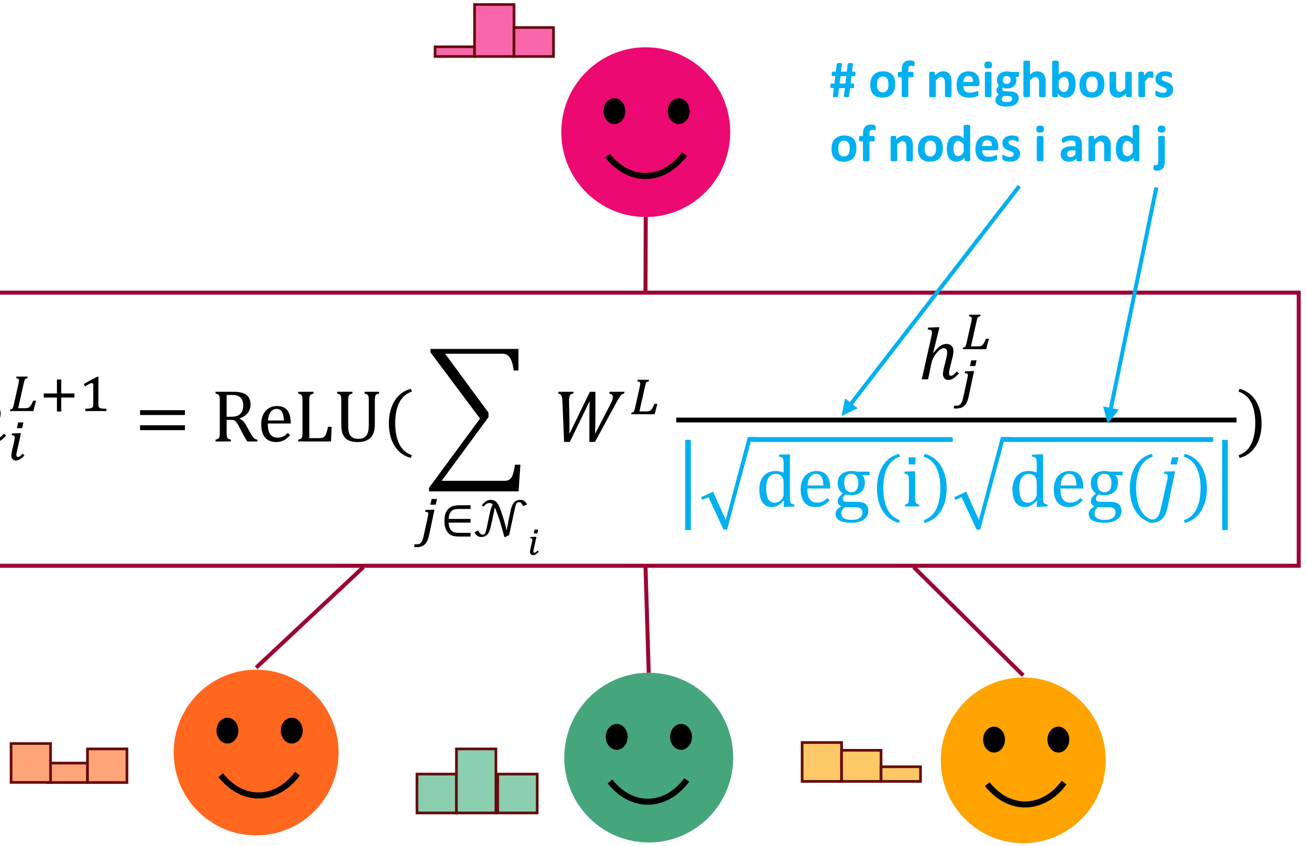
$$h_i^{L+1} = \mathrm{ReLU}\Big(\sum_{j\in\mathcal{N}_i} W^L \frac{h_j^L}{\left|\sqrt{\deg(\mathrm{i})}\sqrt{\deg(j)}\right|}\Big)$$



UNIVERSITY OF CALGARY

# GNN layer example: Graph Convolutional Network

- Graph convolutional network (GCN) learns a simple matrix of weights
  - Can think of this matrix as a 'convolutional kernel' we use at each node instead of pixel

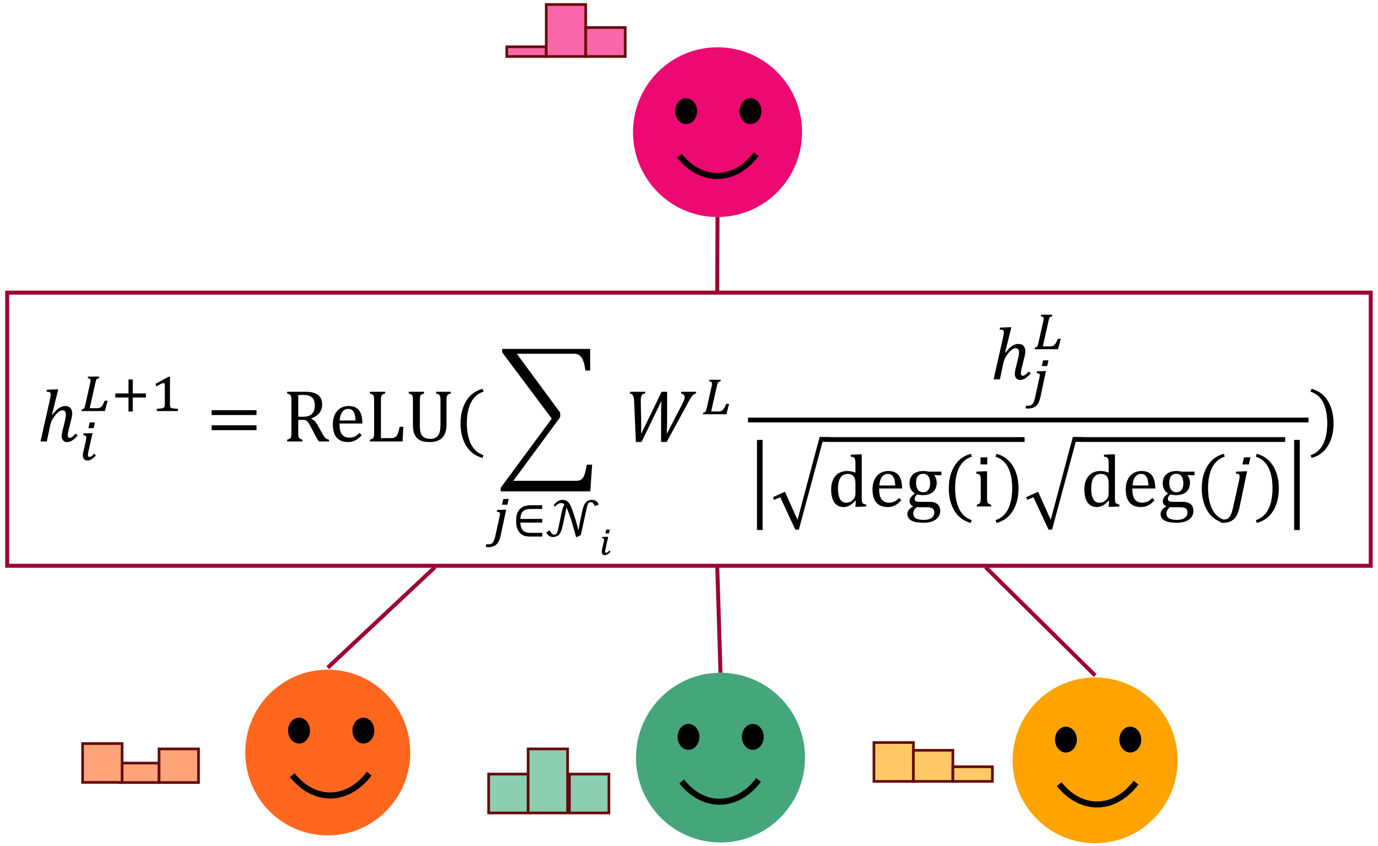



$$h_i^{L+1} = \mathrm{ReLU}\Big(\sum_{j \in \mathcal{N}_i} W^L \frac{h_j^L}{\left|\sqrt{\deg(i)}\sqrt{\deg(j)}\right|}\Big)$$

UNIVERSITY OF CALGARY

# GNN layer example: Graph Convolutional Network

- Graph convolutional network (GCN) learns a simple matrix of weights
  - Can think of this matrix as a ‘convolutional kernel’ we use at each node instead of pixel
- GCN assumes self-edges!
  - So, each node is its own neighbor 🤔



$$h_i^{L+1} = \mathrm{ReLU}\Big(\sum_{j \in \mathcal{N}_i} W^L \frac{h_j^L}{\left|\sqrt{\deg(\mathrm{i})}\sqrt{\deg(j)}\right|}\Big)$$

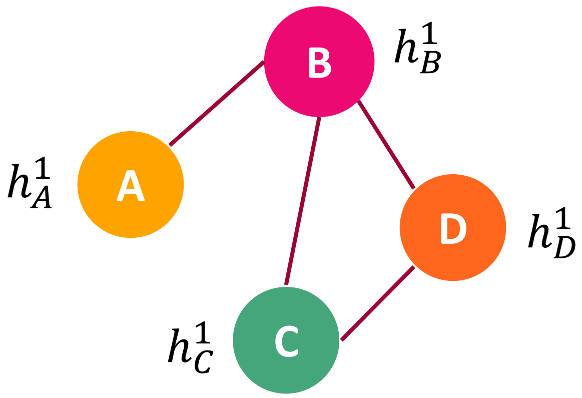


UNIVERSITY OF CALGARY

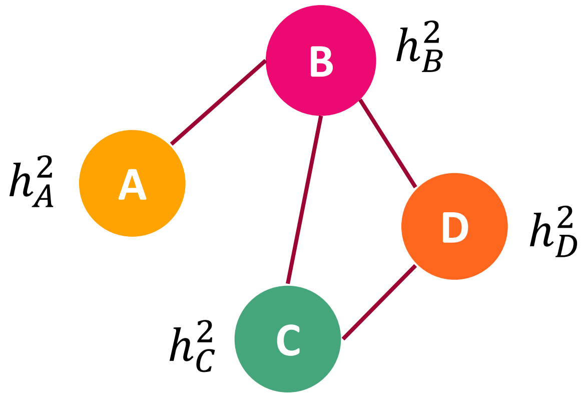
# Putting it all together

- We *could* just learn node/edge/graph embeddings e.g. node2vec
- To tailor network for a specific task, we add an appropriate head
- E.g. Node classification:

Graph layer 1




Graph layer 2




Softmax

*Size of $h_i^2$ = # of classes

| | | |
|---|---|---|
| $h_A^2$ | | [0.7, 0.3] |
| $h_B^2$ | $\dfrac{e^{h_i^2}}{\sum e^{h_i^2}}$ | [0.2, 0.8] |
| $h_C^2$ | | [0.5, 0.5] |
| $h_D^2$ | | [0.1, 0.9] |



UNIVERSITY OF CALGARY

# Summary

- Graph structured data is one of the most general types of data that encompasses a broad range of applications
- Graph neural networks can be used to solve a variety of tasks
- Since graph data is inherently different from other types of data, we need a unique framework to handle it
- Message passing allows for all the information in a graph to be effectively harnessed

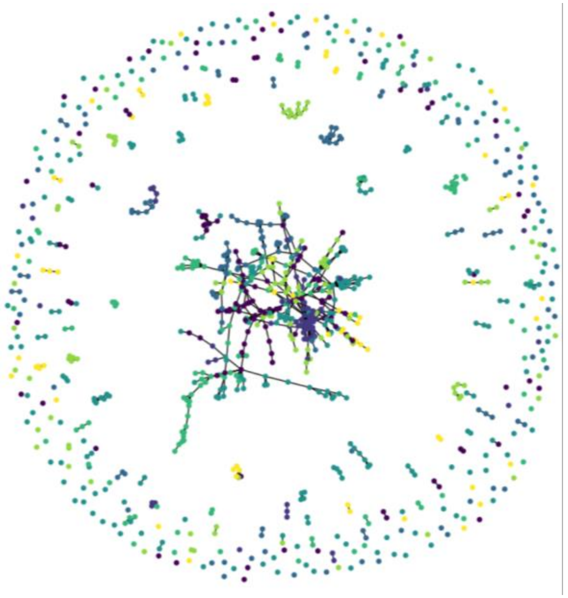


UNIVERSITY OF CALGARY

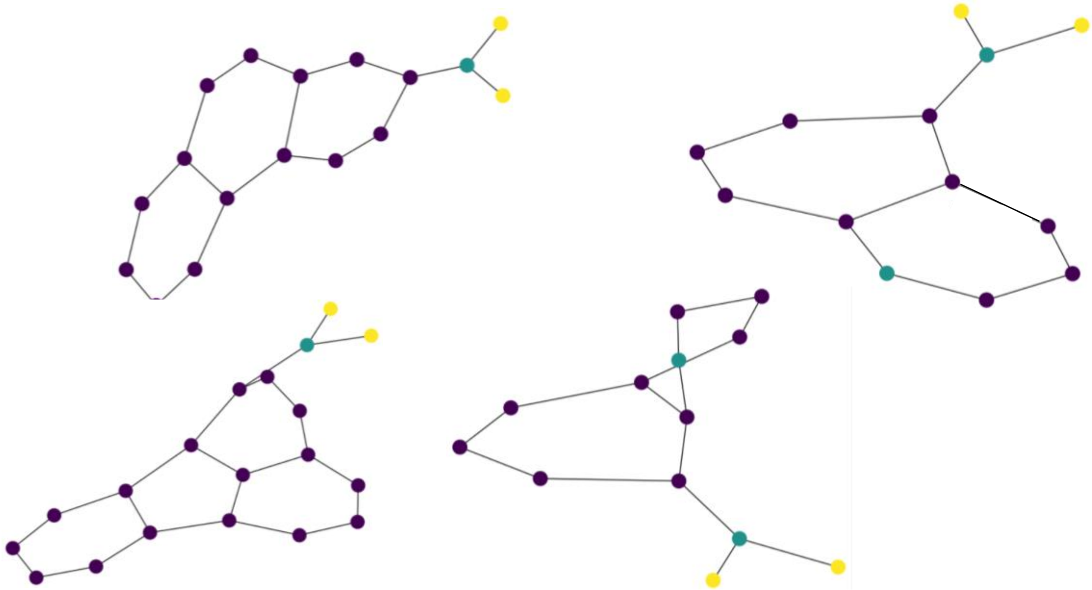
# Next class:

- We will learn how to work with graphs and build models using pytorch-geometric!

**Cora dataset (Citations)**



**TU MUTAG Dataset (Molecules)**





UNIVERSITY OF CALGARY

# References


- J. Leskovec. CS224W: Machine Learning with Graphs. ** Great option if you want a whole COURSE on graph-based machine learning
- W. L. Hamilton. Graph Representation Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3), pages 1-159. 2020.
- Sanchez-Lengeling, et al., "A Gentle Introduction to Graph Neural Networks", Distill, 2021.
- T. N. Kipf et al., Semi-Supervised Classification with graoh Convolutional Networks, ICLR 2017.
- K. Xu et al., How Powerful are Graph Neural Networks? ICLR, 2019.
- W. L. Hamilton et al., Inductive representation Learning on Large Graphs. NIPS, 2017.
- J. Zhou et al., Graph neural networks: A review of methods and applications. AI Open 1. Pages 57-81. 2020.
- Daigavane, et al., "Understanding Convolutions on Graphs", Distill, 2021.
- P. Lippe. Tutorial 7: Graph Neural Networks — UvA DL Notebooks v1.2 documentation (uvadlc-notebooks.readthedocs.io). 2022.
- M. N. Bernstein. Graph convolutional neural networks - Matthew N. Bernstein (mbernste.github.io). 2023.
- Z. Wu et al., A Comprehensive Survey on Graph Neural Networks. IEEE Trans. On Neural networks and Learning Systems. 32(1). 2021.
- R. Anand. Math Behind Graph Neural Networks - Rishabh Anand (rish-16.github.io). 2022.
- D. Grattarola. A practical introduction to GNNs - Part 2 – Daniele Grattarola. 2021.
- E. Benjaminson. Understanding Message Passing in GNNs – Emma Benjaminson – Data Scientist (sassafras13.github.io). 2022.
- T. Masui. Graph Neural Networks with PyG on Node Classification, Link Prediction, and Anomaly Detection | by Tomonori Masui | Towards Data Science. 2022.




UNIVERSITY OF CALGARY

# Thank you!

UNIVERSITY OF CALGARY